Bundle methods for inexact data

Welington de Oliveira and Mikhail Solodov

Abstract Many applications of optimization to real-life problems lead to nonsmooth objective and/or constraint functions that are assessed through "noisy" oracles. In particular, only some approximations to the function and/or subgradient values are available, while exact values are not. For example, this is the typical case in Lagrangian relaxation of large-scale (possibly mixed-integer) optimization problems, in stochastic programming, and in robust optimization, where the oracles perform some *numerical procedure* to evaluate functions and subgradients, such as solving one or more optimization subproblems, multidimensional integration, or simulation. As a consequence, one cannot expect such oracles to provide exact data on the function values and/or subgradients. We review algorithms based on the bundle methodology, mostly developed quite recently, that have the ability to handle inexact data. We adopt an approach which, although not exaustive, covers various classes of bundle methods and various types of inexact oracles, for unconstrained and convexly constrained problems (with both convex and nonconvex objective functions), as well as nonsmooth mixed-integer optimization.

Welington de Oliveira
MINES ParisTech, PSL – Research University, CMA – Centre de Mathématiques Appliquées, Sophia Antipolis, France
e-mail: welington.oliveira@mines-paristech.fr

Mikhail Solodov
IMPA – Instituto de Matemática Pura e Aplicada,
Estrada Dona Castorina 110, Jardim Botânico, Rio de Janeiro, RJ 22460-320, Brazil
e-mail: solodov@impa.br

1 Introduction

Nonsmooth optimization (NSO) appears often in connection with real-life problems that are too difficult to solve directly and need to be decomposed: instead of dealing directly with the difficult problem one may choose (or even have to choose) to solve a sequence of simpler problems (subproblems); see, e.g., [61, 67, 68, 69]. For example, this is a common strategy in stochastic programming [67], in Lagrangian relaxation [10, 44], and in Benders' decomposition [9, Chapter 11]. Lagrangian relaxation leads to nonsmooth convex problems. (Generalized) Benders' decomposition may give rise to constrained nonsmooth and nonconvex optimization problems [20, 55]. In any case, the resulting nonsmooth functions can only be evaluated by an oracle solving (inner) optimization (sub)problems. Solving those subproblems exactly, along many iterations, is at the very least impractical, and is usually not even possible anyway. Similar situations arise when simulation or other numerical procedures are required.

In the NSO setting, the *oracle information* comes in the form of a functional value and *one* subgradient (i.e., the full subdifferential is not available). As is well known, bundle methods, dating back to 1975 (in particular, [43] and [81]), are nowadays among the most efficient algorithms for NSO. Further developments have been in several directions: algorithms with limited memory [39], methods for nonlinearly constrained problems [37, 70], bilevel problems [59, 72], nonmonotone versions [5, 15], second-order methods [52], nonconvex objective functions [22, 28, 35, 51, 57, 77], DC programming [13, 15, 23, 34], convex multiobjetive optimization [54], algorithms for combinatorial and mixed-integer optimization [58, 79], semidefinite programming [19, 30], among others. The original proximal variant of bundle methods has been generalized in [21]. Moreover, other (than proximal) stabilizations have been developed: the level bundle methods proposed in [45], the trust-region variant in [33], the proximal Chebychev center algorithm in [65], and the doubly stabilized bundle method in [14].

Since their invention, and for about 20 years, convergence theory of bundle methods could only handle *exact* oracles, i.e., the exact value of the function and a true subgradient were required, at every point of evaluation. Inexactness was first introduced in [40]; however, approximations of both the function and its subgradients were required to be *asymptotically exact*, i.e., the noise/perturbations had to vanish in the limit. In the context of Lagrangian relaxation, for example, this presumes that we can solve the optimization subproblems with an arbitrarily high precision. While this can be accepted as realistic in some cases, it is certainly not so in general. Next, inexact oracles were considered in [17] for level bundle methods, in [31] for proximal bundle, and in [53] in a special bundle method for the maximal-eigenvalue function. In [17] and [53], the oracle is still asymptotically exact, as in [40]. In [31] it is assumed that the exact value of the function is available, while the subgradient can be computed approximately. The attractive feature of

the analysis in [31] is that, unlike in any previous work on bundle methods, the perturbation in subgradient evaluations need not vanish in the limit. On the other hand, the exact value of the function is still needed, and so this setting is not suitable for some important applications of bundle methods and, in particular, for the Lagrangian relaxation. This is because in that case, evaluating the function and its subgradient is the same task, which amounts to computing a solution of the subproblem (exactly). Nonvanishing perturbations in both the function and subgradient values were introduced in [71]. The next advance, including the idea of *noise attenuation* is [38]. On the side of level bundle variants, [17] was extended in [2, 48, 62].

The more recent developments in handling inexact data made the new variants of bundle methods even more suitable for various real-life applications; see, e.g., [61]. Works such as [29, 62, 63, 78, 79] provide different ways inexact data can be handled by bundle methods, allowing solution of difficult NSO problems (sometimes even exactly, or at least with the desired accuracy, depending on the oracle's assumptions).

In what follows, we discuss various bundle algorithms for variants of the optimization problem

$$\begin{cases} \text{minimize} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in G \ \text{ and } \ h(\mathbf{x}) \leq 0, \end{cases} \tag{1}$$

where $G \subset \mathbb{R}^n$ and the functions f, $h : \mathbb{R}^n \to \mathbb{R}$ are assessed through inexact oracles. Note that in the nonsmooth setting, there is no loss of generality in considering a scalar constraint function h, as it can be defined as the maximum of all the constraint functions. Usually, but not always, f, h and G would be assumed to be convex, and not necessarily all of them would be present in the problem. Specific assumptions would be stated as needed.

The rest of this chapter is organized as follows. We start in Section 2 with defining different types of inexact oracles and presenting some examples of how inexact data appears naturally in applications. By assuming convexity of the involved functions, Section 3 reviews a family of inexact level bundle methods for problem (1) with either convex or nonconvex bounded set G. In Section 4 we consider inexact proximal bundle methods for unconstrained and linearly constrained convex problems. A recent algorithm combining level and proximal ideas is given in Section 5. An inexact proximal bundle method for nonconvex objective functions is discussed in Section 6. Finally, Section 7 contains some concluding remarks and research perspectives.

2 Inexact oracles: the main assumptions and examples

Let the functions f and h in (1) be convex, and assessed via inexact oracles. Specifically, for each given $\mathbf{x} \in \mathbb{R}^n$ an *upper* oracle delivers inexact informa-

tion on f, namely

$$\begin{cases} f_{\mathbf{x}} = f(\mathbf{x}) - \eta^v_{\mathbf{x}} \ \text{ and} \\ \boldsymbol{\xi}_{\mathbf{x}} \in \mathbb{R}^n \ \text{ such that } \ f(\cdot) \geq f_{\mathbf{x}} + \langle \boldsymbol{\xi}_{\mathbf{x}}, \cdot - \mathbf{x} \rangle - \eta^s_{\mathbf{x}} \\ \text{with } \eta^v_{\mathbf{x}} \leq \eta \text{ and } \eta^s_{\mathbf{x}} \leq \eta \text{ for all } \mathbf{x} \in \mathbb{R}^n, \end{cases} \tag{2}$$

and, for the function h, a *lower* oracle provides

$$\begin{cases} h_{\mathbf{x}} = h(\mathbf{x}) - \epsilon_{\mathbf{x}} \ \text{ and} \\ \boldsymbol{\zeta}_{\mathbf{x}} \in \mathbb{R}^n \ \text{ such that } \ h(\cdot) \geq h_{\mathbf{x}} + \langle \boldsymbol{\zeta}_{\mathbf{x}}, \cdot - \mathbf{x} \rangle \\ \text{with } \epsilon_{\mathbf{x}} \leq \epsilon \text{ for all } \mathbf{x} \in \mathbb{R}^n. \end{cases} \tag{3}$$

In the above, the oracles' output is $(f_{\mathbf{x}}, \boldsymbol{\xi}_{\mathbf{x}})$ and $(h_{\mathbf{x}}, \boldsymbol{\zeta}_{\mathbf{x}})$, respectively. The subscripts v and s on the errors in (2) make the distinction between function *value* and *subgradient* errors. The bounds $\eta, \epsilon \geq 0$ on the unknown errors $\eta^v_{\mathbf{x}}$, $\eta^s_{\mathbf{x}}$ and $\epsilon_{\mathbf{x}}$, are possibly unknown as well. However, there are also important situations in which these bounds can actually be chosen by the user and sent, together with $\mathbf{x}$, to the oracles (see Example 2 below).

The exact oracles for f and h correspond to taking $\eta \equiv \epsilon \equiv 0$, and compute $f_{\mathbf{x}} = f(\mathbf{x})$, $h_{\mathbf{x}} = h(\mathbf{x})$, together with their true subgradients. The important subclass of lower oracles returns lower linearizations: $h(\mathbf{x}) - \epsilon_{\mathbf{x}} = h_{\mathbf{x}} \leqslant h(\mathbf{x})$ and $h(\cdot) \geqslant h_{\mathbf{x}} + \langle \boldsymbol{\zeta}_{\mathbf{x}}, \cdot - \mathbf{x} \rangle$. *Upper oracles*, by contrast, can overestimate function values: in (2) $\eta^v_{\mathbf{x}}$ can be negative and $\eta^s_{\mathbf{x}}$ can be positive. We assume the h-oracle to be of lower type for simplicity: as shown in [2, § 5.3], dealing with upper oracles for the constraint function is possible, but requires extra steps in the (level) bundle algorithm. We omit such extra steps to avoid technicalities that may obscure presentation of the main ideas. Interested readers are referred to [2, 3] for a comprehensive treatment of upper oracles for both objective and constraint functions.

We next show how lower and upper oracles appear naturally in practice.

2.1 Examples of inexact oracles

We start with an application which is perhaps the main motivation for investigating algorithms for nonsmooth optimization with inexact data.

Example 1 (Lagrangian relaxation: inexact oracle). Consider the following problem:

$$\begin{cases} \text{maximize} & \varphi(\mathbf{u}) \\ \text{subject to} & \mathbf{u} \in U \ \text{ and } \ c(\mathbf{u}) = \mathbf{0} \in \mathbb{R}^n, \end{cases}$$

where $\varphi : \mathbb{R}^m \to \mathbb{R}$ and $c : \mathbb{R}^m \to \mathbb{R}^n$ are continuous functions, and $U \subset \mathbb{R}^m$ is a nonempty compact set. For a Lagrange multiplier $\mathbf{x} \in \mathbb{R}^n$, the dual

function of the above problem is given by $f(\mathbf{x}) := \max_{\mathbf{u}\in U} L(\mathbf{u},\mathbf{x})$ with $L(\mathbf{u},\mathbf{x}) := \varphi(\mathbf{u}) + \langle \mathbf{x}, c(\mathbf{u})\rangle$. Notice that $L(\mathbf{u},\cdot)$ is convex for any $\mathbf{u}\in U$ fixed.

Given $\mathbf{x}_0$, the oracle (solver) computes a point $\mathbf{u}_0 \in U$ which (approximately) maximizes the function $L(\cdot,\mathbf{x}_0)$ over U, and returns $f_{\mathbf{x}_0} := L(\mathbf{u}_0,\mathbf{x}_0)$ and $\boldsymbol{\xi}_{\mathbf{x}_0} = c(\mathbf{u}_0) \in \partial_{\mathbf{x}} L(\mathbf{u}_0,\mathbf{x}_0)$. Convexity of $L(\mathbf{u}_0,\cdot)$ and the definition of $f(\cdot)$ yield that

$$f(\mathbf{x}) \geq L(\mathbf{u}_0,\mathbf{x}) \geq L(\mathbf{u}_0,\mathbf{x}_0) + \langle \boldsymbol{\xi}_{\mathbf{x}_0}, \mathbf{x}-\mathbf{x}_0\rangle = f_{\mathbf{x}_0} + \langle \boldsymbol{\xi}_{\mathbf{x}_0}, \mathbf{x}-\mathbf{x}_0\rangle\,.$$

Therefore, $f_{\mathbf{x}_0} = L(\mathbf{u}_0,\mathbf{x}_0)$ and $\boldsymbol{\xi}_{\mathbf{x}_0} = c(\mathbf{u}_0)$ satisfy the two first lines in (2) with $\eta^v_{\mathbf{x}_0} = f(\mathbf{x}_0) - L(\mathbf{u}_0,\mathbf{x}_0) \geq 0$ (unknown) and $\eta^s_{\mathbf{x}_0} \equiv 0$, i.e., this is a lower oracle for f. The boundedness assumption $\eta_{\mathbf{x}_0} \leq \eta$ is satisfied if the considered solver is ensured to compute an η-solution $\mathbf{u}_0$ to the subproblem $\max_{\mathbf{u}\in U} L(\mathbf{u},\mathbf{x}_0)$, i.e., a point $\mathbf{u}_0 \in U$ satisfying the condition $\max_{\mathbf{u}\in U} L(\mathbf{u},\mathbf{x}_0) \leq L(\mathbf{u}_0,\mathbf{x}_0) + \eta$. □

More versatile oracles can be obtained if one has control over the solver employed to compute approximate solutions of $\max_{\mathbf{u}\in U} L(\mathbf{u},\mathbf{x})$. This interesting setting is discussed in the next example.

Example 2 (Lagrangian relaxation: on-demand accuracy). In some practical situations (e.g., when $\max_{\mathbf{u}\in U} L(\mathbf{u},\mathbf{x}_0)$ is a combinatorial problem), the error bound η can be chosen and sent to the solver, so that some η-solution $\mathbf{u}_0$ is returned. In other words, the decision-maker can decide to request more or less accuracy from the oracle. Note that requesting high accuracy at unpromising candidate solutions can be a waste of time. More specifically, along the iterative process of minimizing f, the decision-maker may have access to the best estimation $f^{tar} = f_{\mathbf{x}_j}$ of the optimal value. Then, if a new point $\mathbf{x}_i \neq \mathbf{x}_j$ is identified to be a poor solution candidate, the process of computing $f(\mathbf{x}_i)$, i.e., of solving $\max_{\mathbf{u}\in U} L(\mathbf{u},\mathbf{x}_i)$, can be interrupted early (at some point $\mathbf{u}_i \in U$). The oracle delivers the inexact data $f_{\mathbf{x}_i} = L(\mathbf{u}_i,\mathbf{x}_i)$ and $\boldsymbol{\xi}_i = c(\mathbf{u}_i)$ without satisfying the requested tolerance η. This is the case if the inequality $L(\mathbf{u}_i,\mathbf{x}_i) > f^{tar}$ is verified when computing $f(\mathbf{x}_i)$. It is worth mentioning that this kind of test is available in solvers such as Gurobi (www.gurobi.com) and CPLEX (www.cplex.com), for example.

An inexact oracle equipped with such procedures is called *oracle with on-demand accuracy*, introduced in [62] and further investigated in [18] and [78]. □

Another example of how inexact data can arise naturally comes from stochastic programming.

Example 3 (Two-stage stochastic programs). Consider a stochastic program with decision variables organized in two levels, denoted by $\mathbf{x}$ and $\mathbf{y}$ for the first and second stages, respectively. Let $\Omega \subset \mathbb{R}^m$ be a set containing *finitely many* elements (scenarios). If $\boldsymbol{\omega} \in \Omega$ represents uncertainty, for a convex function $\varphi : \mathbb{R}^n \to \mathbb{R}$, vectors $\mathbf{q}(\boldsymbol{\omega})$ and matrices $T(\boldsymbol{\omega})$ and W, the corresponding two-stage program with fixed recourse is

$$\begin{cases} \text{minimize} & \varphi(\mathbf{x}) + \mathbb{E}[\langle \mathbf{q}(\boldsymbol{\omega}), \mathbf{y}\rangle] \\ \text{subject to} & T(\boldsymbol{\omega})\mathbf{x} + W\mathbf{y} = \mathbf{b}(\boldsymbol{\omega}) \quad \forall \boldsymbol{\omega} \in \Omega\,, \\ & \mathbf{x} \in G,\, \mathbf{y} \geqslant 0\,, \end{cases}$$

where $\mathbb{E}[\cdot]$ stands for the expected value with respect to the probability measure on the set Ω. For fixed $\mathbf{x}$ and $\boldsymbol{\omega}$, the *recourse* function

$$Q(\mathbf{x};\boldsymbol{\omega}) := \begin{cases} \text{minimize} & \langle \mathbf{q}(\boldsymbol{\omega}), \mathbf{y}\rangle \\ \text{subject to} & \mathbf{y} \geqslant 0\,, W\mathbf{y} = \mathbf{b}(\boldsymbol{\omega}) - T(\boldsymbol{\omega}) \end{cases}$$

gives in (1) an objective of the form $f(\mathbf{x}) := \varphi(\mathbf{x}) + \mathbb{E}[Q(\mathbf{x};\boldsymbol{\omega})]$, which is finite-valued when the recourse is relatively complete.

For each fixed $\mathbf{x}$ and a given realization $\boldsymbol{\omega}$, the evaluation of the recourse function can be done by solving the dual linear program

$$\begin{cases} \text{maximize} & \langle \mathbf{b}(\boldsymbol{\omega}) - T(\boldsymbol{\omega}), \mathbf{u}\rangle \\ \text{subject to} & W^T\mathbf{u} \leq \mathbf{q}(\boldsymbol{\omega})\,. \end{cases}$$

If, to speed up calculations, instead of performing the maximization for the considered $\boldsymbol{\omega}$ we just take a feasible point $\mathbf{u}_{\mathbf{x},\boldsymbol{\omega}}$ (satisfying $W^T\mathbf{u}_{\mathbf{x},\boldsymbol{\omega}} \leqslant \mathbf{q}(\boldsymbol{\omega})$), then an oracle giving $f_{\mathbf{x}} := \varphi(\mathbf{x}) + \mathbb{E}[\langle \mathbf{b}(\boldsymbol{\omega}) - T(\boldsymbol{\omega}), \mathbf{u}_{\mathbf{x},\boldsymbol{\omega}}\rangle]$, and $\boldsymbol{\xi}_{\mathbf{x}} := \nabla\varphi(\mathbf{x}) - \mathbb{E}[T(\boldsymbol{\omega})^T\mathbf{u}_{\mathbf{x},\boldsymbol{\omega}}]$ is of lower type and fits (2) with $\eta^s_{\mathbf{x}} \equiv 0$.

Alternatively, if we select a (much smaller) subset $\Omega_k \subset \Omega$ of scenarios to perform the subproblem optimization, an upper oracle can be obtained by setting $f_{\mathbf{x}}$ and $\boldsymbol{\xi}_{\mathbf{x}}$ as above but with $\mathbb{E}$ replaced by $\mathbb{E}_k$, the expected value with respect to the probability of (fewer) scenarios in Ω_k. See [64] for more details. □

Nonlinearly constrained variants of stochastic programs arise naturally when one needs to handle risk measures.

Example 4 (CVaR two-stage stochastic programs). With the notation of the previous example, consider the following nonlinearly constrained two-stage stochastic program with parameters $\beta \in (0, 1]$ and $\rho \in \mathbb{R}$:

$$\begin{cases} \text{minimize} & \varphi(\mathbf{x}) + \mathbb{E}[Q(\mathbf{x};\boldsymbol{\omega})] \\ \text{subject to} & \mathbf{x} \in G,\ \mathrm{CVaR}_\beta[Q(\mathbf{x};\boldsymbol{\omega})] \leq \rho\,, \end{cases}$$

where CVaR_β is the *conditional value at risk*. Let $\mathbb{P}(\boldsymbol{\omega})$ be the probability associated to the scenario $\boldsymbol{\omega} \in \Omega$. It can be seen [18] that the convex constraint CVaR_β can be evaluated as the optimal value of the following LP (for $\mathbf{x}$ fixed):

$$\mathrm{CVaR}_\beta[Q(\mathbf{x};\boldsymbol{\omega})] := \begin{cases} \text{maximize} & \sum_{\boldsymbol{\omega}\in\Omega} \pi_{\boldsymbol{\omega}} Q(\mathbf{x};\boldsymbol{\omega}) \\ \text{subject to} & 0 \le \pi_{\boldsymbol{\omega}} \le \mathbb{P}(\boldsymbol{\omega}) \quad \forall \boldsymbol{\omega} \in \Omega \\ & \sum_{\boldsymbol{\omega}\in\Omega} \pi_{\boldsymbol{\omega}} = \beta\,, \end{cases}$$

whose solution can be easily computed by sorting the costs $Q(\mathbf{x};\boldsymbol{\omega})$, $\boldsymbol{\omega} \in \Omega$, and assigning as much as possible weights $\pi_{\boldsymbol{\omega}}$ to the highest costs.

By performing inexact optimization to compute the recourse functions $Q(\mathbf{x};\boldsymbol{\omega})$, one is inexactly evaluating the functions $f(\mathbf{x}) := \varphi(\mathbf{x}) + \mathbb{E}[Q(\mathbf{x};\boldsymbol{\omega})]$ and $h(\mathbf{x}) := \mathrm{CVaR}_\beta[Q(\mathbf{x};\boldsymbol{\omega})] - \rho$. In [18] it is shown how to design efficient inexact oracles for these functions. □

There are applications where the source of inexactness is not in solving optimization subproblems, but is associated to simulation and numerical integration.

Example 5 (Probability maximization problem). Let $(\boldsymbol{\omega}, \Omega, \mathbb{P})$ be a probability space and $c : \mathbb{R}^n \to \mathbb{R}^m$ be a concave mapping. A variant of the so-called *probability maximization problem* is

$$\begin{cases} \text{maximize} & \mathbb{P}[c(\mathbf{x}) \ge \boldsymbol{\omega}] \\ \text{subject to} & \mathbf{x} \in G\,. \end{cases}$$

Problems of this form can be seen as reformulation of chance-constrained optimization problems. They arise, for example, in cascaded-reservoir management [61, 80], financial risk management, and some other areas [11]. If the continuous probability distribution of $\boldsymbol{\omega}$ is log-concave, then the function $f(\mathbf{x}) := -\log(\mathbb{P}[c(\mathbf{x}) \ge \boldsymbol{\omega}])$ is convex (and can be nonsmooth, depending on $\mathbb{P}$ and the covariance matrix of $\boldsymbol{\omega}$). Since a multidimensional integral needs to be numerically computed to evaluate the function, an exact oracle for f is impossible in practice. Inexact values can also be computed by using Monte-Carlo simulation and variance reduction techniques [24]. Such processes induce, in general, upper oracles. Lower estimations of f are possible in some particular cases (for instance when $\boldsymbol{\omega}$ follows a multivariate probability distribution) [76]. It should be noted though that an error bound $\eta > 0$ in (2) can be expected, but cannot be fully guaranteed, as there is always some nonzero probability of a large error showing up (see [3, Lemma 5.1] for details).

If the distribution of $\boldsymbol{\omega}$ is not log-concave, then f is not convex, and algorithms for nonconvex optimization need to be employed. □

The above examples show how inexact oracles arise in connection with practical problems. In the remaining of this chapter, we review several bundle algorithms that are able to handle inexact data.

3 Inexact level bundle methods

Throughout this section we make the assumption that f and h in (1) are convex functions, and that inexact oracles satisfying (2) and (3) are available. We also assume that G is a compact set, but not necessary a convex one.

A versatile family of bundle algorithms, called *level bundle* methods, was introduced in [40, 45]. As will be seen in what follows, simple level bundle variants can handle both exact and inexact data in the same fashion, without any special treatment or modifications in the case of inexact oracles. This is different from proximal bundle methods, which require modifications in the inexact setting.

3.1 Models, localizer sets, and optimality certificate

Observe that the two first lines in (2) and (3) yield

$$\begin{aligned} f(\cdot) &\geq f(\mathbf{x}) + \langle \boldsymbol{\xi}_{\mathbf{x}}, \cdot - \mathbf{x} \rangle - (\eta^v_{\mathbf{x}} + \eta^s_{\mathbf{x}}) \,, \\ h(\cdot) &\geq h(\mathbf{x}) + \langle \boldsymbol{\zeta}_{\mathbf{x}}, \cdot - \mathbf{x} \rangle - \epsilon_{\mathbf{x}} \,, \end{aligned} \tag{4}$$

from which, evaluating at $\mathbf{x}$ we deduce that $\eta^v_{\mathbf{x}} + \eta^s_{\mathbf{x}} \geq 0$ (regardless of the signs of the individual error terms) and $\epsilon_{\mathbf{x}} \geq 0$. As a result,

$$\boldsymbol{\xi}_{\mathbf{x}} \in \partial_{(\eta^v_{\mathbf{x}} + \eta^s_{\mathbf{x}})} f(\mathbf{x}) \quad \text{with} \quad \eta^v_{\mathbf{x}} + \eta^s_{\mathbf{x}} \geq 0 \quad \text{for all} \;\; \mathbf{x} \in \mathbb{R}^n \,, \tag{5}$$

$$\boldsymbol{\zeta}_{\mathbf{x}} \in \partial_{(\epsilon_{\mathbf{x}})} f(\mathbf{x}) \quad \text{with} \quad \epsilon_{\mathbf{x}} \geq 0 \quad \text{for all} \;\; \mathbf{x} \in \mathbb{R}^n \,. \tag{6}$$

Even if in (2) and (3) the values of the bounds for errors η and ϵ are unknown, the inequalities above imply that $\eta \geq \eta^v_{\mathbf{x}} \geq -\eta^s_{\mathbf{x}} \geq -\eta$ and $\epsilon \geq \epsilon_{\mathbf{x}} \geq 0$. Thus, oracle errors are bounded from below (by $-\eta$ and 0, respectively).

Let k be the index of the current iteration. Having called the oracle at previous iterates $\mathbf{x}_j$, bundle algorithms accumulate linearizations of the functions to construct their cutting-plane models:

$$\hat{f}_k(\cdot) := \max_{j \in \mathcal{J}_k} \{ f_{\mathbf{x}_j} + \langle \boldsymbol{\xi}_{\mathbf{x}_j}, \cdot - \mathbf{x}_j \rangle \} \leq f(\cdot) + \eta \tag{7}$$

$$\hat{h}_k(\cdot) := \max_{j \in \mathcal{J}_k} \{ h_{\mathbf{x}_j} + \langle \boldsymbol{\zeta}_{\mathbf{x}_j}, \cdot - \mathbf{x}_j \rangle \} \leq h(\cdot), \tag{8}$$

where $\mathcal{J}_k$ is an index set (typically $\mathcal{J}_k \subset \{1, \ldots, k\}$), and the inequalities follow from the second lines in (2) and (3). With these models and an additional parameter $f^{lev}_k \in \mathbb{R}$, we can define the following *localizer set*:

$$\text{Ł}_k := \{ \mathbf{x} \in G \mid \hat{f}_k(\mathbf{x}) \leq f^{lev}_k, \; \hat{h}_k(\mathbf{x}) \leq 0 \} \,. \tag{9}$$

When the constraint function h is not present, the localizer set becomes the level set of the cutting-model $\hat{f}_k$, justifying thus the name "*level bundle methods*". The following is a simple but useful result that will allow to determine approximate lower bounds for the optimal value f^* of problem (1).

Lemma 1. *If* $\text{Ł}_k = \emptyset$, *then* $f_k^{lev} \leq f^* + \eta$.

Proof. It follows from (7) and (8) that Ł_k approximates the level set of f over the feasible set in the following sense:

$$\{\mathbf{x} \in G \mid f(\mathbf{x}) \leq f_k^{lev} - \eta,\, h(\mathbf{x}) \leqslant 0\} \subset \text{Ł}_k\,.$$

Hence, if Ł_k is an empty set, then so is $\{\mathbf{x} \in G \mid f(\mathbf{x}) \leq f_k^{lev} - \eta,\, h(\mathbf{x}) \leqslant 0\}$. The feasible set $\{\mathbf{x} \in G \mid h(\mathbf{x}) \leqslant 0\}$ is nonempty, by the standing assumption. As a result, $f_k^{lev} - \eta$ must be a lower bound for f^*. □

Let f_0^{low} be a given lower bound for the optimal value f^*. A handy manner to update such a bound along the iterative process is to set $f_{k+1}^{low} := f_k^{lev}$ if $\text{Ł}_k = \emptyset$, and $f_{k+1}^{low} := f_k^{low}$ otherwise. With this rule, Lemma 1 ensures that $f_k^{low} \leq f^* + \eta$ for all k. We can thus define the useful *improvement function*

$$F_j(f_k^{low}) := \max\{f_{\mathbf{x}_j} - f_k^{low},\, h_{\mathbf{x}_j}\} \quad \text{and} \quad \Delta_k := \min_{j \leq k} F_j(f_k^{low})\,. \tag{10}$$

Remark 1 (Approximate optimality test). Note that if $F^j(f_k^{low}) \leq \delta^{tol}$ for some index $j \leq k$ and the given tolerance $\delta^{tol} \geq 0$ (which is the case when $\Delta_k \leq \delta^{tol}$), then $h_{\mathbf{x}_j} \leq \delta^{tol}$ and $f_{\mathbf{x}_j} \leq f_k^{low} + \delta^{tol}$. By using the oracles' assumptions we get $h(\mathbf{x}_j) - \epsilon \leq h_{\mathbf{x}_j} \leq \delta^{tol}$ and $f(\mathbf{x}_j) - \eta \leq f_{\mathbf{x}_j} \leq f_k^{low} + \delta^{tol} \leq f^* + \eta + \delta^{tol}$, i.e.,

$$h(\mathbf{x}_j) \leq \epsilon + \delta^{tol} \quad \text{and} \quad f(\mathbf{x}_j) \leq f^* + 2\eta + \delta^{tol}\,.$$

In other words, $\mathbf{x}_j$ is a $(\max\{2\eta,\, \epsilon\} + \delta^{tol})$-solution to problem (1). We can thus employ the value Δ_k as an (approximate) optimality certificate: if it holds that $\Delta_k \leq \delta^{tol}$, then $\mathbf{x}^{best} := \mathbf{x}_j$ for j yielding the minimum in (10) is an approximate solution in the given sense. □

3.2 An algorithmic pattern

We now present an inexact level bundle method for the class of problems in the form (1), having convex functions f and h, and a compact set G.

Algorithm 1 is a simplistic version of [2, Algorithm 1]: it does not provide either an implementable rule to define $x_{k+1} \in \text{Ł}_k$ nor a scheme for keeping the size of the bundle $\mathcal{J}_k$ bounded (limited memory). We shall return to these more specific issues a bit later, after proving convergence of the method to $(\max\{2\eta, \epsilon\} + \delta^{tol})$-solution of problem (1).

Algorithm 1: Inexact level bundle method: an algorithmic pattern

Data: Parameters $\gamma \in (0,1)$, $\delta^{tol} \geq 0$, and a lower bound f_0^{low} for (1).
Step 0 (*Initialization*) Choose a starting point $\mathbf{x}_0 \in G$ and call the oracles to compute $(f_{\mathbf{x}_0}, \boldsymbol{\xi}_{\mathbf{x}_0})$ and $(h_{\mathbf{x}_0}, \boldsymbol{\zeta}_{\mathbf{x}_0})$. Set $\mathcal{J}_0 := \{0\}$ and $k := 0$.
Step 1 (*Stopping test*) Compute Δ_k as in (10). If $\Delta_k \leq \delta^{tol}$, then stop and return $\mathbf{x}^{best} := \mathbf{x}_j$ for j yielding the minimum in (10).
Step 2 (*Next iterate*) Set $f_k^{lev} := f_k^{low} + \gamma\Delta_k$ and define the localizer set Ł_k as in (9). If $\text{Ł}_k = \emptyset$, then define $f_{k+1}^{low} := f_k^{lev}$ and go to Step 4. Otherwise, set $f_{k+1}^{low} := f_k^{low}$ and choose $\mathbf{x}_{k+1} \in \text{Ł}_k$.
Step 3 (*Oracle call*) Compute the data $(f_{\mathbf{x}_{k+1}}, \boldsymbol{\xi}_{\mathbf{x}_{k+1}})$ and $(h_{\mathbf{x}_{k+1}}, \boldsymbol{\zeta}_{\mathbf{x}_{k+1}})$.
Step 4 (*Bundle management*) Set $\mathcal{J}_{k+1} := \mathcal{J}_k \cup \{k+1\}$ if $\mathbf{x}_{k+1}$ is available and $\mathcal{J}_{k+1} := \mathcal{J}_k$ otherwise. Set $k := k+1$ and go back to Step 1.

Note that the algorithm defines the next trial point in the localizer set Ł_k. Hence, for all $j \in \mathcal{J}_k$, we have that: (i) $f_{\mathbf{x}_j} + \langle \boldsymbol{\xi}_{\mathbf{x}_j}, \mathbf{x}_{k+1} - \mathbf{x}_j \rangle \leq f_k^{lev}$ and (ii) $h_{\mathbf{x}_j} + \langle \boldsymbol{\zeta}_{\mathbf{x}_j}, \mathbf{x}_{k+1} - \mathbf{x}_j \rangle \leq 0$. Using the Cauchy-Schwarz inequality in (i), we obtain that $f_{\mathbf{x}_j} - f_k^{lev} \leq \|\boldsymbol{\xi}_{\mathbf{x}_j}\| \|\mathbf{x}_{k+1} - \mathbf{x}_j\| \leq \|\Lambda_f\| \|\mathbf{x}_{k+1} - \mathbf{x}_j\|$, whereas inequality (ii) yields $h_{\mathbf{x}_j} \leq \|\boldsymbol{\zeta}_{\mathbf{x}_j}\| \|\mathbf{x}_{k+1} - \mathbf{x}_j\| \leq \|\Lambda_h\| \|\mathbf{x}_{k+1} - \mathbf{x}_j\|$. The existence of finite constants Λ_f and Λ_h above is ensured by [32, Proposition 6.2.2], because the oracle errors are bounded, (5) and (6) hold, and G is a bounded set. By taking $\Lambda := \max\{\Lambda_f, \Lambda_h\}$ we have thus shown that

$$\Delta_k \leq F_j(f_k^{low}) = \max\{f_{\mathbf{x}_j} - f_k^{lev}, h_{\mathbf{x}_j}\} \leq \Lambda \|\mathbf{x}_{k+1} - \mathbf{x}_j\| \quad \forall j \in \mathcal{J}_k \,. \quad (11)$$

Theorem 1. *For problem* (1)*, assume that f and h are convex, $G \neq \emptyset$ is compact, and that the inexact oracles satisfy* (2) *and* (3)*. If $\delta^{tol} > 0$, then after finitely many steps Algorithm 1 stops with a* $(\max\{2\eta, \epsilon\} + \delta^{tol})$*-solution to problem* (1)*.*

Proof. Suppose that Algorithm 1 does not terminate. In particular, this implies that $\Delta_k > \delta^{tol} > 0$ for all k. Then, using also (11), we obtain that $0 < \delta^{tol} \leq \Lambda \|\mathbf{x}_{k+1} - \mathbf{x}_j\|$ for all k and all $j \leq k$. This contradicts the fact that the bounded sequence $(\mathbf{x}_k) \subset G$ has a convergent subsequence. Therefore, the inequality $\Delta_k \leq \delta^{tol}$ must hold after finitely many steps. Remark 1 then concludes the proof. □

As long as $\mathbf{x}_{k+1} \in \text{Ł}_k$ and $\mathcal{J}_k = \{1, 2, \ldots, k\}$, the above result ensures convergence (in the given sense) of Algorithm 1 for any nonempty compact set G, regardless of its structure. For instance, G can be a mixed-integer set, or even something more complicated. If one can either verify that Ł_k is empty or, otherwise, pick *any* point in Ł_k in some practical/implementable manner, Algorithm 1 is an appealing general strategy, due to its reliability and simplicity.

3.3 Exploiting the domain

Let $\hat{\mathbf{x}}_k$ be the stability center, for instance: the last iterate $\mathbf{x}_{k-1}$, the best candidate $\mathbf{x}^{best}$, or another point chosen by some rule. Then, determining the next trial point $\mathbf{x}_{k+1}$ in $Ł_k$ (given in (9)) can be done by minimizing the *distance* to the stability center $\hat{\mathbf{x}}_k$, i.e., $\mathbf{x}_{k+1}$ solves

$$\begin{cases} \text{minimize} & d(\mathbf{x}, \hat{\mathbf{x}}_k) \\ \text{subject to} & \mathbf{x} \in Ł_k \,. \end{cases} \tag{12}$$

Appropriate choices for the distance function $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_+$ and for the stability center $\hat{\mathbf{x}}_k$ depend on the structure of G. In what follows, we discuss some possible variants of Algorithm 1 by exploiting this structure.

3.3.1 The combinatorial setting

Let G be composed of binary variables: $G = \{\mathbf{x} \in \{0,\, 1\}^n :\, A\mathbf{x} \leq b\}$, for some given matrix $A \in \mathbb{R}^{m\times n}$ and vector $b \in \mathbb{R}^m$. Taking $d(\mathbf{x}, \hat{\mathbf{x}}_k) := \frac{1}{2}\|\mathbf{x} - \hat{\mathbf{x}}_k\|_2^2$, we get that $d(\mathbf{x}, \hat{\mathbf{x}}_k) = \frac{1}{2}\sum_{i=1}^n \mathbf{x}_i - \sum_{i=1}^n \mathbf{x}_i\hat{\mathbf{x}}_i + \frac{1}{2}\sum_{i=1}^n \hat{\mathbf{x}}_i$. Hence, the master problem (12) boils down to the binary LP:

$$\begin{cases} \text{minimize} & \langle \frac{1}{2}\mathbf{1} - \hat{\mathbf{x}}_k, \mathbf{x}\rangle \\ \text{subject to} & f_{\mathbf{x}_j} + \langle \boldsymbol{\xi}_{\mathbf{x}_j}, \mathbf{x} - \mathbf{x}_j\rangle \leq f_k^{lev}, \quad j \in \mathcal{J}_k \\ & h_{\mathbf{x}_j} + \langle \boldsymbol{\zeta}_{\mathbf{x}_j}, \mathbf{x} - \mathbf{x}_j\rangle \leq 0, \quad j \in \mathcal{J}_k \\ & A\mathbf{x} \leq b,\ \mathbf{x} \in \{0,\, 1\}^n. \end{cases}$$

We point out that one does not need to solve this subproblem up to optimality at every iteration of Algorithm 1: merely a feasible point, or a certificate that the feasible set is empty, is enough to ensure convergence of the level bundle method. That said, solving up to optimality may decrease the number of oracle calls, as reported in [79].

3.3.2 The mixed-integer setting

Suppose that G is the mixed-integer set $G = \{\mathbf{x} = (\mathbf{x}^c, \mathbf{x}^i) \in \mathbb{R}^{n_c} \times \mathbb{Z}^{n_i} :\, A\mathbf{x} \leq b\}$. The work [58] proposes to define the next trial point as (approximate) solution of the following mixed-integer master problem:

$$\begin{cases} \text{minimize} & \|\mathbf{x}-\hat{\mathbf{x}}_k\|_\diamond \\ \text{subject to} & f_{\mathbf{x}_j}+\langle \boldsymbol{\xi}_{\mathbf{x}_j}, \mathbf{x}-\mathbf{x}_j\rangle \le f_k^{lev}, \quad j\in\mathcal{J}_k \\ & h_{\mathbf{x}_j}+\langle \boldsymbol{\zeta}_{\mathbf{x}_j}, \mathbf{x}-\mathbf{x}_j\rangle \le 0, \quad j\in\mathcal{J}_k \\ & A\mathbf{x}\le b,\ \mathbf{x}=(\mathbf{x}^c,\mathbf{x}^i)\in\mathbb{R}^{n_c}\times\mathbb{Z}^{n_i}\,, \end{cases} \tag{13}$$

where $d(\mathbf{x},\hat{\mathbf{x}}_k)=\|\mathbf{x}-\hat{\mathbf{x}}_k\|_\diamond$, and $\|\cdot\|_\diamond$ is a given norm, for instance the ℓ_1, ℓ_∞ or ℓ_2 norms. For the two first choices, (13) becomes a MILP, whereas for $\|\cdot\|_\diamond=\|\cdot\|_2^2$ (13) is MIQP. For these choices good numerical results were reported in [58] for dealing with convex MINLP problems: when compared to classical methods for this type of problems, the given level bundle methods could reduce the number of oracle calls in around 22%, which can yield a significant CPU time reduction depending on the computational cost of the oracles.

3.3.3 The convex setting

Let now G be a closed convex set, the more friendly setting considered in most bundle methods in the literature [2, 17, 40, 45, 62]. We stick with the classical choice for the stability function $d(\mathbf{x},\hat{\mathbf{x}}_k)=\frac{1}{2}\|\mathbf{x}-\hat{\mathbf{x}}_k\|_2^2$, yielding the following QP master problem:

$$\begin{cases} \text{minimize} & \frac{1}{2}\|\mathbf{x}-\hat{\mathbf{x}}_k\|_2^2 \\ \text{subject to} & f_{\mathbf{x}_j}+\langle \boldsymbol{\xi}_{\mathbf{x}_j}, \mathbf{x}-\mathbf{x}_j\rangle \le f_k^{lev}, \quad j\in\mathcal{J}_k \\ & h_{\mathbf{x}_j}+\langle \boldsymbol{\zeta}_{\mathbf{x}_j}, \mathbf{x}-\mathbf{x}_j\rangle \le 0, \quad j\in\mathcal{J}_k \\ & x\in G\,. \end{cases} \tag{14}$$

The next iterate is therefore the projection of the stability center onto the localizer set. The projection provides some useful inequalities that permit to analyze the complexity of Algorithm 1. To this end, let us consider special iterations called "critical iterations", the ones which make significant progress in the quest of solving (1) or improving the current lower bound f_k^{low}. More specifically, critical iterations would be indexed by ℓ and are defined as follows: set $\Delta_{k(0)}=\Delta_0$, $\ell=0$ and update

$$\ell:=\ell+1,\ k(\ell):=k \quad \text{whenever} \quad \Delta_k\le(1-\gamma)\Delta_{k(\ell)} \ \text{ or } \ \mathrm{L}_k=\emptyset\,.$$

We can thus split the iterative process of Algorithm 1 into cycles: denote the ℓ-th cycle as $\mathcal{K}^\ell:=\{k(\ell),k(\ell)+1,\ldots,k(\ell+1)-1\}$.

Proposition 1. *Consider Algorithm 1 and assume that G is a compact convex set with diameter $D<\infty$ and $\hat{\mathbf{x}}_k=\hat{\mathbf{x}}\in G$ is fixed for all $k\in\mathcal{K}^\ell$. Then, any iteration index $k\in\mathcal{K}^\ell$ with $\Delta_k>\delta^{tol}>0$ may differ no more from $k(\ell)$ than the following bound:*

$$k - k(\ell) + 1 \leq \left(\frac{\Lambda D}{(1-\gamma)\Delta_k} \right)^2 .$$

Proof. It follows from the definition of the cycle $\mathcal{K}^\ell$ that $f_k^{low} = f_{k(\ell)}^{low}$ and Δ_k is nonincreasing for all $k \in \mathcal{K}^\ell$. This shows that f_k^{lev} is nonincreasing as well. Hence, $\text{Ł}_k \subset \text{Ł}_{k-1}$ for all $k, k-1 \in \mathcal{K}^\ell$ (because the bundle $\mathcal{J}_k$ keeps all linearizations). Consider the iteration indexed by $k-1 \in \mathcal{K}^\ell$. The projection of $\hat{\mathbf{x}}$ onto Ł_{k-1} yields $\mathbf{x}_k$ and, moreover, the inequality

$$\langle \hat{\mathbf{x}} - \mathbf{x}_k, \mathbf{x}_k - \mathbf{x} \rangle \geq 0 \quad \forall\, \mathbf{x} \in \text{Ł}_{k-1} .$$

As $\mathbf{x}_{k+1} \in \text{Ł}_k \subset \text{Ł}_{k-1}$, the inequality $\langle \hat{\mathbf{x}} - \mathbf{x}_k, \mathbf{x}_k - \mathbf{x}_{k+1} \rangle \geq 0$ used in $\|\mathbf{x}_{k+1} - \hat{\mathbf{x}}\|_2^2 = \|\mathbf{x}_{k+1} - \mathbf{x}_k + (\mathbf{x}_k - \hat{\mathbf{x}})\|_2^2$ gives

$$\|\mathbf{x}_{k+1} - \hat{\mathbf{x}}\|_2^2 \geq \|\mathbf{x}_k - \hat{\mathbf{x}}\|_2^2 + \|\mathbf{x}_{k+1} - \mathbf{x}_k\|_2^2 .$$

By employing (11) with $j = k$ and recalling that $\Delta_k > (1-\gamma)\Delta_{k(\ell)}$ by the definition of $k(\ell)$, we obtain that

$$\begin{aligned} D^2 \geq \|\mathbf{x}_{k+1} - \hat{\mathbf{x}}\|_2^2 &\geq \|\mathbf{x}_k - \hat{\mathbf{x}}\|_2^2 + \left(\frac{\Delta_k}{\Lambda} \right)^2 \\ &> \|\mathbf{x}_k - \hat{\mathbf{x}}\|_2^2 + \left(\frac{(1-\gamma)\Delta_{k(\ell)}}{\Lambda} \right)^2 . \end{aligned}$$

The result then follows by some simple manipulations of the above relation; see [2, Lemma 4] for more details. □

We note that the hypothesis $\hat{\mathbf{x}} \in G$ is only used to ensure that the relation $D^2 \geq \|\mathbf{x}_{k+1} - \hat{\mathbf{x}}\|_2^2$ holds. In fact, we could use any (non-feasible) stability center $\hat{\mathbf{x}}$: since G is compact, the distance $\|\mathbf{x}_{k+1} - \hat{\mathbf{x}}\|_2^2$ would be bounded by another constant, say $\tilde{D}^2$, and the given complexity analysis would hold with D replaced with $\tilde{D}$. We next estimate the maximum number of oracle calls to obtain $\Delta_k \leq \delta^{tol}$.

Theorem 2. *Under the assumptions of Proposition 1, assume further that $\delta^{tol} > 0$ in Algorithm 1. Then, to reach an optimality measure Δ_k smaller than $\delta^{tol} > 0$ Algorithm 1 performs at most*

$$\left(1 + \frac{f^* + \eta - f_0^{low}}{\gamma \delta^{tol}} \right) \left(\frac{\Lambda D}{(1-\gamma)\delta^{tol}} \right)^2 \quad \textit{iterations.}$$

Proof. Notice that every time Ł_k is empty the lower bound f_k^{low} of $f^* + \eta$ is increased by an amount of $\gamma \Delta_k$ ($> \gamma \delta^{tol}$). Since f_0^{low} is finite, the maximum number of cycles ℓ^{mx} times the stepsize $\gamma \delta^{tol}$ is less than $f^* + \eta - f_0^{low}$, i.e., $\ell^{mx} \leqslant (f^* + \eta - f_0^{low})/(\gamma \delta^{tol})$. The result then follows from Proposition 1 by noting that each cycle $\mathcal{K}^\ell$ has at most $(\Lambda D)^2/((1-\gamma)\delta^{tol})^2$ iterations. See [2, Theorem 2] for more details. □

Proposition 1 requires the bundle of information $\mathcal{J}_k$ to keep all the linearizations indexed by $k \in \mathcal{K}^\ell$. However, the bundle can be managed arbitrarily at critical iterations $k(\ell)$: for instance, we could simply empty $\mathcal{J}_k$ by keeping only the new linearization indexed by $k(\ell)$. We mention that for a limited-memory variant of Algorithm 1, one needs to include the so-called aggregate linearizations, as is standard in modern bundle methods; see, for instance, Step 7 of [2, Algorithm 1].

3.4 Handling oracles with on-demand accuracy

We highlight that Algorithm 1 does not make any special treatment of inexact data: it works exactly the same way for both exact and inexact oracles. More efficient variants are obtained if we can control the oracles' inexactness. We now address this subject by focusing on oracles with *on-demand accuracy*.

Following [18], we say that $\mathbf{x}_k \in G$ is an *f-substantial iterate* if the inexact value of the function $f_{\mathbf{x}_k}$ meets the descent target $f_k^{tar} \in \mathbb{R}$. Similarly, $\mathbf{x}_k \in G$ is said to be an *h-substantial iterate* if the inexact constraint value $h_{\mathbf{x}_k}$ meets the feasibility target $h_k^{tar} \in \mathbb{R}$. An iterate $\mathbf{x}_k$ is said to be *substantial* if it is both f- and h-substantial. Moreover, we denote with $\mathcal{S}$ the index set gathering iterations that provided substantial iterates. We will refer to $\mathcal{S}$ as the *substantial set*. The aim of this section is to make a few changes in Algorithm 1, in order to deal with lower oracles with asymptotically vanishing errors for substantial iterates. Such oracles are referred to as lower oracles with on-demand accuracy [2]. They satisfy (2) and (3) with the following additional assumptions, where $0 \leq \eta_k \leq \eta$, $f_k^{tar} \in \mathbb{R}$ and $h_k^{tar} \in \mathbb{R}$ are given parameters:

$$\begin{cases} \eta^s_{\mathbf{x}_k} \equiv 0 \quad \text{(lower oracle)} \\ \eta^v_{\mathbf{x}_k} \leq \eta_k \quad \text{if } f_{\mathbf{x}_k} \leq f_k^{tar} \\ \epsilon^v_{\mathbf{x}_k} \leq \eta_k \quad \text{if } h_{\mathbf{x}_k} \leq h_k^{tar}. \end{cases} \tag{15}$$

This kind of oracles provide information with accuracy up to η_k for substantial iterates.

We emphasize that the assumptions (2) and (3) satisfying also (15) are still quite general and cover many situations of interest. For given $\mathbf{x}_k$ and $\eta_k = 0$, the oracle provides exact information if both $f_{\mathbf{x}_k}$ and $h_{\mathbf{x}_k}$ meet the targets. Moreover, if both targets f_k^{tar} and h_k^{tar} are set as $+\infty$ for all iterations and $\eta_k = 0$, then the oracles are exact. *Asymptotically exact* versions of these oracles are obtained if $f_k^{tar} = h_k^{tar} = +\infty$ for all k and $\eta_k \to 0$. Combinations between partially exact and asymptotically exact versions are also possible. For this setting, in order to get an exact solution to problem (1) the algorithm must force $\eta_k \to 0$ (only) for substantial iterates.

Let f_k^{low} be a proven lower bound for f^*, the optimal value of (1). At any substantial iterate $j \in \mathcal{S}$, the oracle bound η_j is known and can be exploited in the definition of the improvement function. We therefore define

$\Delta_k := \min_{j \leq k} F_j^{\mathcal{S}}(f_k^{low})$ with

$$F_j^{\mathcal{S}}(f_k^{low}) := \begin{cases} \max\{f_{\mathbf{x}_j} - f_k^{low},\, h_{\mathbf{x}_j}\} + \eta_j & \text{if } j \in \mathcal{S}, \\ +\infty & \text{otherwise.} \end{cases} \tag{16}$$

Due to the *on-demand accuracy* assumption, Lemma 9 in [2] ensures that $F_j^{\mathcal{S}}(f_k^{low}) \geq \max\{f(\mathbf{x}_j) - f_k^{low},\, h(\mathbf{x}_j)\} \geq 0$. Hence, if $F_j^{\mathcal{S}}(f_k^{low}) = 0$ then $\mathbf{x}_j$ is an optimal solution to problem (1) and $f_k^{low} = f^*$. In this case $\eta_j = 0$, i.e., the oracle is exact at the substantial iterate $\mathbf{x}_j$.

In order to obtain $\Delta_k \to 0$, it is necessary to force η_k to zero for substantial iterates. This can be done by controlling the oracle error in Step 3 of Algorithm 1 as follows:

Step 3' (*On-demand accuracy*)

- Step 3.1 (*Controlling the error*) Update the oracle error by setting $\eta_{k+1} = \theta\Delta_{k(\ell)}$, for a given $\theta \in (0, (1-\gamma)^2)$ (here $k(\ell)$ the last critical iteration).
- Step 3.2 (*Target Updating*) Set $f_{k+1}^{tar} = f_k^{lev} + (1-\gamma)\Delta_k$ and $h_{k+1}^{tar} = (1-\gamma)\Delta_k$.
- Step 3.3 (*Oracle call*) Compute the data $(f_{\mathbf{x}_{k+1}}, \boldsymbol{\xi}_{\mathbf{x}_{k+1}})$ and $(h_{\mathbf{x}_{k+1}}, \boldsymbol{\zeta}_{\mathbf{x}_{k+1}})$ satisfying (2), (3) and (15).

We have the following result, whose proof can be found in [2, Theorem 6].

Theorem 3. *Consider Algorithm 1 with $\delta^{tol} = 0$, Step 3 replaced by Step 3' above, and optimality certificate given in* (16)*. Let $\mathbf{x}_k^{best}$ be the point defining the value of Δ_k. Then $\Delta_k \to 0$ and every accumulation point of the sequence $(\mathbf{x}_k^{best})$ is an exact solution to problem* (1)*.*

The interest of using oracles with on-demand accuracy is evident: the algorithm can compute an exact solution to the problem, even though most of its iterations are inexact. The main idea is that there is no gain in "wasting" time for computing exact data at non-promising points (whose function values are greater than the specified targets f_k^{tar} and h_k^{tar}). As shown by the numerical experience in [2,18,62], for a large battery of stochastic programs (two-stage, risk-free, risk-averse and chance-constrained ones) it is possible to save up to 79% of the total computational time (while still finding an exact solution) just by letting the bundle method control how accurate the oracle information should be at substantial iterates.

In the linearly-constrained setting, the complexity analysis of level bundle methods with on-demand accuracy is very similar to the one of exact bundle methods, which is known for being optimal or nearly-optimal [8,59]. We refer interested readers to [62, Section 3.2.3] for a formal statement.

We finalize this section by mentioning that there exist specialized inexact bundle methods dealing with unbounded set G; see [62, Section 4] for oracles with on-demand accuracy and [48] for the more general setting. However, none of these works handle nonlinearly constrained problems. Nonlinearly

constrained problems with unbounded set G are considered in [3], but in a proximal bundle framework.

4 Inexact proximal bundle methods

For the sake of simplicity, in most part of this section we drop the nonlinear constraint h in (1) and focus on the simpler setting of

$$\begin{cases} \text{minimize} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in G, \end{cases} \tag{17}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function and $G \subset \mathbb{R}^n$ is a nonempty convex and closed set, typically polyhedral. The nonlinearly constrained setting will be discussed in Subsection 4.4.3.

To deal with problem (17) we now consider proximal bundle methods [33,43]. As in the level variant, proximal bundle algorithms define new iterates by making use of a cutting-plane model $\hat{f}_k$ for f, a stability center $\hat{\mathbf{x}}_k \in G$ and a stabilization function d. For the classical choice $d(\mathbf{x}, \hat{\mathbf{x}}_k) = \frac{1}{2t_k}\|\mathbf{x} - \hat{\mathbf{x}}_k\|_2^2$, with $t_k > 0$ a prox-parameter, the new iterate $\mathbf{x}_{k+1}$ of a proximal bundle algorithm is the solution of the master problem

$$\begin{cases} \text{minimize} & \hat{f}_k(\mathbf{x}) + \frac{1}{2t_k}\|\mathbf{x} - \hat{\mathbf{x}}_k\|_2^2 \\ \text{subject to} & \mathbf{x} \in G\,. \end{cases} \tag{18}$$

As is well known, if G is polyhedral, then (18) is equivalent to solving the following QP:

$$\begin{cases} \text{minimize} & r + \frac{1}{2t_k}\|\mathbf{x} - \hat{\mathbf{x}}_k\|_2^2 \\ \text{subject to} & f_{\mathbf{x}_j} + \langle \boldsymbol{\xi}_{\mathbf{x}_j}, \mathbf{x} - \mathbf{x}_j \rangle \le r, \quad j \in \mathcal{J}_k, \\ & \mathbf{x} \in G,\ r \in \mathbb{R}\,. \end{cases}$$

It is worth to mention that for the same model $\hat{f}_k$ and the same stability center $\hat{\mathbf{x}}_k$, one can find the proximal and level parameters t_k and f_k^{lev} such that (12) (without the nonlinear constraint h) and (18) generate the same next iterate $\mathbf{x}_{k+1}$. In this formal theoretical sense, the level and proximal approaches can be considered equivalent. But the details of the implementation and practical performance can be quite different. In particular, the key parameters are updated by strategies which are specific to each of the methods, and thus the generated iterates are not the same in practice.

4.1 Descent test and optimality certificate

The stability center in (inexact) proximal bundle methods is usually updated according to the following *descent rule*. Let

$$v_k := f_{\hat{\mathbf{x}}_k} - \hat{f}_k(\mathbf{x}_{k+1}), \tag{19}$$

and let $\kappa \in (0,1)$ be a parameter.

$$\text{If } f_{\mathbf{x}_{k+1}} \leq f_{\hat{\mathbf{x}}_k} - \kappa v_k, \text{ then } \hat{\mathbf{x}}_{k+1} := \mathbf{x}_{k+1}; \text{ otherwise } \hat{\mathbf{x}}_{k+1} := \hat{\mathbf{x}}_k\,. \tag{20}$$

Some other descent rules allowing more flexibility in defining stability centers (or providing strong convergence in general Hilbert spaces) can be found in [5, 21, 78] and [1], respectively.

Note first that in the setting of inexact oracles one can have $v_k < 0$ in (19), which renders the descent test (20) meaningless. In the convex case, this situation can only be caused by the oracle inexactness. The method then checks whether v_k is (or is not) sufficiently positive, to detect when inaccuracy is becoming excessive/cumbersome. If v_k is not (sufficiently) positive, the method increases the prox-parameter t_k. The motivation for this is as follows. If v_k is not positive, (19) suggests that $\mathbf{x}_{k+1}$ does not sufficiently minimize the model $\hat{f}_k(\cdot)$. Increasing t_k in (18) decreases the weight of the quadratic term therein, thus increasing the relative weight of $\hat{f}_k(\cdot)$ in the minimization problem. This has the effect of decreasing the value of the model at candidate points, eventually (once t_k is large enough) making v_k in (19) acceptably positive. Once the latter is achieved, the iteration proceeds as in a standard bundle method. This procedure is called *noise attenuation* [38].

In what follows we provide some useful ingredients to check inexact optimality and to keep the bundle of information $\mathcal{J}_{k+1}$ limited. Assuming G is polyhedral, or that an appropriate constraint qualification [73] holds in (18), the optimality conditions for (18) yield that

$$\mathbf{x}_{k+1} := \hat{\mathbf{x}}_k - t_k\hat{\boldsymbol{\xi}}_k, \quad \text{with} \quad \hat{\boldsymbol{\xi}}_k := \mathbf{p}_k^f + \mathbf{p}_k^G,$$

where $\mathbf{p}_k^f \in \partial\hat{f}_k(\mathbf{x}_{k+1})$ and $\mathbf{p}_k^G \in \partial i_G(\mathbf{x}_{k+1})$, i_G being the indicator function of the set G. Making use of Lagrange multipliers α_j^k associated to the constraints $f_{\mathbf{x}_j} + \langle \boldsymbol{\xi}_{\mathbf{x}_j}, \mathbf{x}_{k+1} - \mathbf{x}_j\rangle \leq r$, $j \in \mathcal{J}_k$, it further holds that

$$\mathbf{p}_k^f := \sum_{j\in\mathcal{J}_k} \alpha_j^k \boldsymbol{\xi}_{\mathbf{x}_j},\ \sum_{j\in\mathcal{J}_k} \alpha_j^k = 1,\ \alpha_j^k \geq 0,\ j \in \mathcal{J}_k\,.$$

Such multipliers can be used to save storage without impairing convergence. More precisely, "inactive" indices, corresponding to $\alpha_j^k = 0$, can be dropped for the next iteration: $\mathcal{J}_{k+1} \supset \{j \in \mathcal{J}_k : \alpha_j^k \neq 0\}$. An even more economical model can be constructed using the so-called *aggregate linearization*

$$\bar{f}_{k_a}(\mathbf{x}) := \hat{f}_k(\mathbf{x}_{k+1}) + \langle \hat{\boldsymbol{\xi}}_k, \mathbf{x} - \mathbf{x}_{k+1} \rangle,$$

which satisfies $\bar{f}_{k_a}(\mathbf{x}) \leq f_k(\mathbf{x}) + i_G(\mathbf{x}) \leq f(\mathbf{x}) + \eta + i_G(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^n$, by the oracle assumptions in (2) and the definition of $\hat{\boldsymbol{\xi}}_k$. As in the exact proximal bundle methods, this linearization plays an important role in the bundle management: it can be shown that if the new cutting-plane model is defined by the *minimal bundle* $\mathcal{J}_{k+1} := \{k+1, k_a\}$, this is still enough to ensure convergence.

The following result establishes a helpful connection between the predicted decrease $v_k := f_{\hat{\mathbf{x}}_k} - \hat{f}_k(\mathbf{x}_{k+1})$ and the *aggregate error* $\hat{e}_k := f_{\hat{\mathbf{x}}_k} - \bar{f}_{k_a}(\hat{\mathbf{x}}_k)$. In particular, it gives a certificate of (approximate) optimality to problem (17).

Lemma 2. *It holds that* $\hat{e}_k \geqslant -2\eta$, $v_k = \hat{e}_k + t_k \|\hat{\boldsymbol{\xi}}_k\|_2^2$, *and*

$$\hat{\boldsymbol{\xi}}_k \in \partial_{(\hat{e}_k + 2\eta)}[f(\hat{\mathbf{x}}_k) + i_G(\hat{\mathbf{x}}_k)].$$

Proof. It follows from the definitions of $\hat{e}_k$, v_k, $\mathbf{x}_{k+1}$ and $\bar{f}_{k_a}$ that

$$\hat{e}_k = f_{\hat{\mathbf{x}}_k} - \bar{f}_{k_a}(\hat{\mathbf{x}}_k) = f_{\hat{\mathbf{x}}_k} - [\hat{f}_k(\mathbf{x}_{k+1}) + \langle \hat{\boldsymbol{\xi}}_k, \hat{\mathbf{x}}_k - \mathbf{x}_{k+1} \rangle] = v_k - t_k \|\hat{\boldsymbol{\xi}}_k\|_2^2.$$

In addition, the oracle definition (2) and inequality $\bar{f}_{k_a}(\mathbf{x}) \leq f(\mathbf{x}) + \eta$ for all $\mathbf{x} \in G$ give: $\hat{e}_k = f_{\hat{\mathbf{x}}_k} - \bar{f}_{k_a}(\hat{\mathbf{x}}_k) \geq (f(\hat{\mathbf{x}}_k) - \eta) - (f(\hat{\mathbf{x}}_k) + \eta) = -2\eta$.

Let $\mathbf{x} \in \mathbb{R}^n$ be an arbitrary point. Proposition 5 and the oracle assumptions (2) yield

$$\begin{aligned}
\eta + f(\mathbf{x}) + i_G(\mathbf{x}) &\geq \bar{f}_{k_a}(\mathbf{x}) = \hat{f}_k(\mathbf{x}_{k+1}) + \langle \hat{\boldsymbol{\xi}}_k, \mathbf{x} - \mathbf{x}_{k+1} \rangle \\
&= f_{\hat{\mathbf{x}}_k} - (f_{\hat{\mathbf{x}}_k} - \hat{f}_k(\mathbf{x}_{k+1})) + \langle \hat{\boldsymbol{\xi}}_k, \mathbf{x} - \hat{\mathbf{x}}_k \rangle + \langle \hat{\boldsymbol{\xi}}_k, \hat{\mathbf{x}}_k - \mathbf{x}_{k+1} \rangle \\
&= f_{\hat{\mathbf{x}}_k} - (\hat{e}_k + t_k \|\hat{\boldsymbol{\xi}}_k\|_2^2) + \langle \hat{\boldsymbol{\xi}}_k, \mathbf{x} - \hat{\mathbf{x}}_k \rangle + \langle \hat{\boldsymbol{\xi}}_k, t_k \hat{\boldsymbol{\xi}}_k \rangle \\
&\geq f(\hat{\mathbf{x}}_k) - \eta - \hat{e}_k + \langle \hat{\boldsymbol{\xi}}_k, \mathbf{x} - \hat{\mathbf{x}}_k \rangle .
\end{aligned}$$

Since $i_G(\hat{\mathbf{x}}_k) = 0$, we conclude that $\hat{\boldsymbol{\xi}}_k \in \partial_{(\hat{e}_k + 2\eta)}[f(\hat{\mathbf{x}}_k) + i_G(\hat{\mathbf{x}}_k)]$. □

As a consequence of Lemma 2, if for some infinite index set $\mathcal{K} \subset \{0, 1, 2, \ldots\}$ we have

$$\limsup_{\mathcal{K} \ni k \to \infty} \hat{e}_k \leq 0 \quad \text{and} \quad \lim_{\mathcal{K} \ni k \to \infty} \|\hat{\boldsymbol{\xi}}_k\|_2 = 0, \tag{21}$$

then every accumulation point $\bar{\mathbf{x}}$ of the sequence $(\hat{\mathbf{x}}_k \mid k \in \mathcal{K})$ satisfies $0 \in \partial_{2\eta}[f(\bar{\mathbf{x}}) + i_G(\bar{\mathbf{x}})]$, i.e., $\bar{\mathbf{x}}$ is a 2η-solution to (17). This gives the needed (approximate) optimality measures.

4.2 Inexact proximal bundle algorithm

We now present an inexact proximal bundle method for convex problems in the form (17). We refer to [38] for the original algorithm and to [63] for further enhancements.

Algorithm 2: Inexact Proximal Bundle Method

Data: Stopping tolerances $\delta_1^{tol}, \delta_2^{tol} \geq 0$, a descent parameter $\kappa \in (0,1)$, a stepsize $t_0 \geq \underline{t} > 0$, and $\tau \in [\frac{1}{2}, 1)$.

Step 0 (*Initialization*) Choose a starting point $\mathbf{x}_0 \in G$, set $\hat{\mathbf{x}}_0 := \mathbf{x}_0$, and call the oracle to compute $(f_{\mathbf{x}_0}, \boldsymbol{\xi}_{\mathbf{x}_0})$. Set $\mathcal{J}_0 := \{0\}$, $k := 0$ and $\mathtt{na} := 0$.

Step 1 (*Trial point computation*) Find $\mathbf{x}_{k+1}$ solving (18) and let α^k denote the corresponding optimal simplicial multiplier. Compute $\hat{\boldsymbol{\xi}}_k = (\hat{\mathbf{x}}_k - \mathbf{x}_{k+1})/t_k$, $v_k = f_{\hat{\mathbf{x}}_k} - \hat{f}_k(\mathbf{x}_{k+1})$, and $\hat{e}_k = v_k - t_k\|\hat{\boldsymbol{\xi}}_k\|_2^2$.

Step 2 (Stopping criterion) If $\|\hat{\boldsymbol{\xi}}_k\|_2 \leq \delta_1^{tol}$ and $\hat{e}_k \leq \delta_2^{tol}$, stop.

Step 3 (Inaccuracy detection) If $\hat{e}_k < -\tau t_k\|\hat{\boldsymbol{\xi}}_k\|_2^2$, set $t_k := 10t_k$, $\mathtt{na} := k$ and loop back to step 1.

Step 4 (Oracle call and descent test) Call the inexact oracle (2) to compute $(f_{\mathbf{x}_{k+1}}, \boldsymbol{\xi}_{\mathbf{x}_{k+1}})$. If the descent test (20) holds, then declare the iterate serious: set $\hat{\mathbf{x}}_{k+1} := \mathbf{x}_{k+1}$, $\mathtt{na} := 0$. Otherwise, declare the iterate null: set $\hat{x}_{k+1} := \hat{x}_k$.

Step 5 (Bundle management) Choose $\mathcal{J}_{k+1} \supseteq \{j \in \mathcal{J}^k : \alpha_j^k \neq 0\} \cup \{k+1\}$. (Another possibility is $\mathcal{J}_{k+1} := \{k+1, k_a\}$.)

Step 6 (Stepsize updating and loop) If the iterate was declared serious, select $t_{k+1} \geq t_k$. If the iterate was declared null, either set $t_{k+1} := t_k$, or choose $t_{k+1} \in [0.1t_k, t_k]$ such that $t_{k+1} \geq \underline{t}$ if $\mathtt{na} = 0$. Increase k by 1 and go to Step 1.

In the exact setting, the aggregate error $\hat{e}_k$ is always nonnegative. This implies the inequality $v_k = \hat{e}_k + t_k\|\hat{\boldsymbol{\xi}}_k\|_2^2 \geq t_k\|\hat{\boldsymbol{\xi}}_k\|_2^2 = \frac{1}{t_k}\|\mathbf{x}_{k+1} - \hat{\mathbf{x}}_k\|_2^2$. The inaccuracy detection inequality $\hat{e}_k < -\tau t_k\|\hat{\boldsymbol{\xi}}_k\|_2^2$ at Step 3 implies that the oracle error is too excessive: note that in this case

$$v_k = \hat{e}_k + t_k\|\hat{\boldsymbol{\xi}}_k\|_2^2 < (1-\tau)t_k\|\hat{\boldsymbol{\xi}}_k\|_2^2 = \frac{1-\tau}{t_k}\|\mathbf{x}_{k+1} - \hat{\mathbf{x}}_k\|_2^2,$$

i.e., the predicted decrease is not sufficiently positive. The role of increasing t_k in this situation had been already discussed above.

4.3 Convergence analysis

Throughout this section, we consider Algorithm 2 with $\delta_1^{tol} = \delta_2^{tol} = 0$, applied to problem (17), assumed to have a solution. The oracle satisfies (2).

We start the analysis considering the infinite noise attenuation loop.

Proposition 2 (Infinite noise attenuation loop). *Suppose the algorithm loops indefinitely between Steps 1 and 3, after the last serious iterate $\bar{\mathbf{x}} = \hat{\mathbf{x}}_{k_0}$ is generated. Then* (21) *holds with $\mathcal{K} = \{0, 1, 2, \ldots\}$ and $\bar{\mathbf{x}}$ is a 2η-solution to problem* (17)*.*

Proof. Since the algorithm loops indefinitely between Steps 1 and 3, we have that $\hat{e}_k < -\tau t_k\|\hat{\boldsymbol{\xi}}_k\|_2^2$ for all k large enough. Then, Lemma 2 ensures that

$-2\eta \leq \hat{e}_k < -\tau t_k \|\hat{\boldsymbol{\xi}}_k\|_2^2 < 0$ for k large enough. Letting $k \to \infty$ in this relation, and recalling that $t_k \to +\infty$ by Step 3 of the algorithm, we conclude that (21) holds. □

Next, we proceed as usual in proximal bundle methods, with two possibilities: (i) the algorithm generates an infinite sequence of null steps after a last serious iterate; (ii) an infinite sequence of serious steps is produced. Note that $\hat{e}_k \geq -\tau t_k \|\hat{\boldsymbol{\xi}}_k\|_2^2$ for all k yielding either a null or a serious step (as otherwise the step is that of noise attenuation), and in particular $v_k \geq 0$ for such iterations.

The next result addresses item (i). It makes use of the crucial inequality

$$0 \geq \limsup_{k\to\infty} \left[f_{\mathbf{x}_k} - \hat{f}_{k-1}(\mathbf{x}_k) \right], \tag{22}$$

whose proof can be consulted in [38, Lemma 3.3, Eq. (3.6)] or [63, Theorem 6.4 and Section 7.3].

Proposition 3 (Infinite loop of null steps). *If the algorithm generates an infinite sequence of null steps after the last serious iterate $\bar{\mathbf{x}} = \hat{\mathbf{x}}_{k_0}$ is obtained, then* (21) *holds with $\mathcal{K} = \{0, 1, 2, \ldots\}$ and $\bar{\mathbf{x}}$ is a 2η-solution to problem* (17).

Proof. In this case, the descent test (20) never holds for k large enough, i.e., $f_{\mathbf{x}_{k+1}} > f_{\hat{\mathbf{x}}_{k_0}} - \kappa v_k$. We then obtain that

$$\begin{aligned} f_{\mathbf{x}_{k+1}} - \hat{f}_k(\mathbf{x}_{k+1}) &= f_{\mathbf{x}_{k+1}} - f_{\hat{\mathbf{x}}_{k_0}} + (f_{\hat{\mathbf{x}}_{k_0}} - \hat{f}_k(\mathbf{x}_{k+1})) \\ &\geq -\kappa v_k + (f_{\hat{\mathbf{x}}_{k_0}} - \hat{f}_k(\mathbf{x}_{k+1})) = (1-\kappa) v_k \,, \end{aligned}$$

where $v_k \geq 0$ for the type of iterations in considertion. Using (22), we conclude that $v_k \to 0$. As $v_k = \hat{e}_k + t_k \|\hat{\boldsymbol{\xi}}_k\|_2^2$, $\hat{e}_k \geq -\tau t_k \|\hat{\boldsymbol{\xi}}_k\|_2^2$ in this case and $t_k \geq \underline{t} > 0$, the relations (21) follow. □

We now consider the sequence of serious steps.

Proposition 4 (Infinite number of serious steps). *If the algorithm generates an infinite sequence of serious steps, then the relations in* (21) *hold for $\mathcal{K} = \{k \in \mathbb{N} \mid$ the descent test* (20) *is satisfied for $k\}$, and every accumulation point of $(\hat{\mathbf{x}}_k \mid k \in \mathcal{K})$ is a 2η-solution to problem* (17).

Proof. Let $f^* > -\infty$ be the optimal value of (17). Then the oracle ensures that $f_{\mathbf{x}_k} \geq f^* - \eta$ for all k. For $k \in \mathcal{K}$, it holds that $f_{\hat{\mathbf{x}}_{k+1}} \leq f_{\hat{\mathbf{x}}_k} - \kappa v_k$, where $v_k \geq 0$. Then for any $k \in \mathcal{K}$, we obtain that

$$\infty > f_{\hat{\mathbf{x}}_0} - (f^* - \eta) \geq f_{\hat{\mathbf{x}}_0} - f_{\hat{\mathbf{x}}_{k+1}} = \sum_{m=0,\, m\in\mathcal{K}}^{k} [f_{\hat{\mathbf{x}}_m} - f_{\hat{\mathbf{x}}_{m+1}}] \geq \kappa \sum_{m=0,\, m\in\mathcal{K}}^{k} v_m.$$

Letting in the above $k \to \infty$ implies that $\sum_{k=0,\, k\in\mathcal{K}}^{\infty} v_k$ is finite, and hence, $v_k \to 0$ as $\mathcal{K} \ni k \to \infty$. The rest of the reasoning is the same as in the end of the proof of Proposition 3. □

Combining Propositions 2, 3 and 4, convergence analysis of Algorithm 2 is now summarized as follows.

Theorem 4. *Suppose that the inexact oracle satisfies* (2)*, and that in Algorithm 2 one sets* $\delta_1^{tol} = \delta_2^{tol} = 0$*. Then the optimality certificate* (21) *holds for some index set* $\mathcal{K}$*, and every accumulation point* $\bar{\mathbf{x}}$ *of the sequence* $(\hat{\mathbf{x}}_k \mid k \in \mathcal{K})$ *is a* 2η*-solution to problem* (17)*.*

We remark that $(\hat{\mathbf{x}}_k)$ is guaranteed to be bounded (and thus have accumulation points) if f has bounded level sets. Indeed, since v_k is nonnegative at serious steps, $f_{\hat{\mathbf{x}}_k} \le f_{\hat{\mathbf{x}}_0} \le f(\hat{x}_0) + \eta$ for all k. This shows that $f(\hat{\mathbf{x}}_k) \le f(\hat{x}_0) + 2\eta$ for all k.

On the other hand, Theorem 4.5 in [63] does not even assume that the problem has a solution to prove that $\limsup_{\mathcal{K}\ni k\to\infty} f_{\hat{\mathbf{x}}_k} \le \inf_{\mathbf{x}\in G} f(\mathbf{x}) + \eta$. For this, the stopping test of Algorithm 2 has to be changed to the following: stop when $\|\hat{\boldsymbol{\xi}}_k\|$ and $\max\{0, \hat{e}_k + \langle \hat{\boldsymbol{\xi}}_k, \hat{\mathbf{x}}_k\rangle\}$ are small enough. Employing the latter quantities is useful for proving convergence without imposing boundedness on the sequence of serious steps; see [63, Section 4.2] for details.

4.4 Inexact proximal bundle method variants

As for the level bundle method, the proximal bundle algorithm also has many variants as discussed, for example, in [63, Section 7]. Below we list some of them. But first, let us note that with exact oracles, Algorithm 2 becomes one of the usual exact proximal bundle methods. In particular, the noise attenuation procedure in Step 3 is never triggered (because the aggregate error is always nonnegative in the exact convex case). Then, Theorem 4 holds with $\eta = 0$.

4.4.1 Partially inexact proximal bundle method

Let the inexact oracle satisfy (2) and (15) with $f_k^{tar} := f_{\hat{x}_k} - \kappa v_k$. Then, Algorithm 2 becomes a partially inexact method, in which at serious iterates evaluations are by an oracle with on-demand accuracy, but null iterates are of the general inexact type. It can be shown that in this setting the algorithm computes an exact solution to problem (17), see [63, Section 7.1].

4.4.2 Incremental proximal bundle method

Suppose that the objective function in (17) has the additive structure, i.e., $f(\mathbf{x}) := \sum_{i=1}^{m} f^i(\mathbf{x})$, with $f^i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$, being convex functions. For example, this is the case in Lagrangian relaxation of multistage stochastic integer programming, where every component function f^i is the optimal value of a mixed-integer suproblem

$$f^i(\mathbf{x}) := \begin{cases} \text{maximize} & \varphi^i(\mathbf{u}) + \langle \mathbf{x}, c^i(\mathbf{u}) \rangle \\ \text{subject to} & \mathbf{u} \in U^i \subset \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2}\,. \end{cases} \tag{23}$$

We assume that it is possible for any $\mathbf{x}$ and $\eta > 0$ to find η-solutions $\mathbf{u}^i_{\mathbf{x}}$ to the problems (23), that is,

$$\begin{aligned} &\mathbf{u}^i_{\mathbf{x}} \in U^i \;:\; \varphi^i(\mathbf{u}^i_{\mathbf{x}}) + \langle \mathbf{x}, c^i(\mathbf{u}^i_{\mathbf{x}}) \rangle \geq f^i(\mathbf{x}) - \eta, \\ &\text{yielding } \; f^i_{\mathbf{x}} = \varphi^i(\mathbf{u}^i_{\mathbf{x}}) \quad \text{and} \quad \boldsymbol{\xi}^i_{\mathbf{x}} := c^i(\mathbf{u}^i_{\mathbf{x}})\,. \end{aligned} \tag{24}$$

However, the computational effort required may be substantial and will depend on the choice of η. The number of subproblems m may be large, adding further difficulty to the oracle. It may be possible to simultaneously carry out more than one such calculation using parallel computing.

In the context in consideration, the only favorable properties of f are its convexity and the additive structure. *Incremental bundle methods* [16,60,78] take advantage of this structure using (possibly inexact) information about some of the functions f^i and their subgradients, but trying to avoid evaluating all the m functions (i.e., solving all the m subproblems (23)), at least at certain "unpromising" trial points.

At the k-th iteration of the algorithm, having generated trial points $\mathbf{x}_1, \ldots, \mathbf{x}_k$ and the corresponding lower-oracle pairs $(f^i_{\mathbf{x}_j}, \boldsymbol{\xi}^i_{\mathbf{x}_j})$ as defined in (24), the *disaggregate* model for f is obtained as follows:

$$f^{\boldsymbol{m}}_k(\mathbf{x}) := \sum_{i=1}^{m} \hat{f}^i_k(\mathbf{x}) \quad \text{with} \quad \hat{f}^i_k(\mathbf{x}) := \max_{j \in \mathcal{J}^i_k} \{ f^i_{\mathbf{x}_j} + \langle \boldsymbol{\xi}^i_{\mathbf{x}_j}, \mathbf{x} - \mathbf{x}_j \rangle \} \tag{25}$$

for some $\mathcal{J}^i_k \subseteq \{1, \ldots, k\}$. Since the oracle is of the lower type, it can be shown (see, e.g., [60, Lemma 2.3]) that if $\mathcal{J}^i_k \supset \mathcal{J}_k$ for all $i = 1, \ldots, m$, then $\hat{f}_k(\mathbf{x}) \leq f^{\boldsymbol{m}}_k(\mathbf{x}) \leq f(\mathbf{x})$ for all $\mathbf{x}$, i.e., the inexact disaggregate model $f^{\boldsymbol{m}}_k$ is a better approximation of f than $\hat{f}_k$.

Aside from offering greater accuracy, the disaggregate model has another advantage, which has already been exploited in [16]: it allows for "partial" oracle evaluations in which information about the value of f at a new trial point $\mathbf{x}_{k+1}$ may be obtained without calling all the m individual subproblem oracles (24). Specifically, for any $I_{k+1} \subseteq \{1, \ldots, m\}$, we have

$$f_{I_{k+1}} := \underbrace{\sum_{i\in I_{k+1}} f^i_{\mathbf{x}_{k+1}}}_{|I_{k+1}| \text{ oracle calls}} + \underbrace{\sum_{j\notin I_{k+1}} \hat{f}^j_k(\mathbf{x}_{k+1})}_{\substack{m-|I_{k+1}| \text{ model} \\ \text{approximations}}} \leq f(\mathbf{x}_{k+1})\,. \tag{26}$$

The inequality (26) holds because $f^i_{\mathbf{x}_{k+1}} \leq f^i(\mathbf{x}_{k+1})$ and $\hat{f}^i_k(\mathbf{x}_{k+1}) \leq f^i(\mathbf{x}_{k+1})$ for $i = 1, \ldots, m$. Note that the component functions f^j with $j \notin I_{k+1}$ are not accessed by the oracles.

When examining an iterate $\mathbf{x}_{k+1}$, the method will sometimes not require the exact value $f(\mathbf{x}_{k+1})$, and will be able to discard the point based only on the information that $f(\mathbf{x}_{k+1}) > f^{tar}_k$ for some "target" value. If we can check this condition using a bound of the form $f_{I_{k+1}}$ as defined in (26), we may be able to save on oracle calls by calling $|I_{k+1}| < m$ subproblem oracles at $\mathbf{x}_{k+1}$, instead of all the m oracles. The pseudo-code below outlines a class of procedures for attaining this goal by incrementally enlarging the set I_{k+1}, as needed.

Algorithm 3: Managing oracle calls

Input: a point $\mathbf{x}_{k+1} \in G$, a target value $f^{tar}_k \in \mathbb{R}$, bounds η^i_{k+1}, $i = 1, \ldots, m$ for the oracle errors.
Initialize $I_{k+1} \leftarrow \emptyset$ and $f_{I_{k+1}} := \check{f}_k(\mathbf{x}_{k+1})$
while $|I_{k+1}| < m$ *and* $f_{I_{k+1}} \leq f^{tar}_k$ **do**
  Select some nonempty $J \subseteq \{1, \ldots, m\} \backslash I_{k+1}$
  for $i \in J$ **do**
    Call oracle (24) to compute $(f^i_{\mathbf{x}_{k+1}}, \boldsymbol{\xi}^i_{\mathbf{x}_{k+1}})$ with accuracy $\eta^i_{k+1} \geq 0$
  Set $I_{k+1} := I_{k+1} \cup J$ and compute $f_{I_{k+1}}$ from (26)
Set $f_{\mathbf{x}_{k+1}} := f_{I_{k+1}}$ and return I_{k+1}, $f_{\mathbf{x}_{k+1}}$, and $\{(f^i_{\mathbf{x}_{k+1}}, \boldsymbol{\xi}^i_{\mathbf{x}_{k+1}})\}_{i\in I_{k+1}}$

Initially, the algorithm above approximates the value $f(\mathbf{x}_{k+1})$ by using the cutting-plane models $\hat{f}^i$, $i = 1, \ldots, m$. It then selects a set of subproblems $J \subseteq \{1, \ldots, m\}$ and calls their oracles. Next, it includes J into the set I_{k+1} and updates its estimate $f_{I_{k+1}}$ of $f(\mathbf{x}_{k+1})$. If this calculation is sufficient to establish that $f(\mathbf{x}_{k+1}) > f^{tar}_k$, or already all subproblem oracles have been called, it exits. Otherwise, it repeats this procedure with another subset J of oracles.

In serial computing, it would be most economical to fix $|J| = 1$ and thus evaluate a single subproblem oracle per iteration. In a parallel setting, it might be advantageous to choose $|J| > 1$, and perform the oracle calls concurrently.

Note that Algorithm 3 is not specific about how to select the set J of subproblem oracles to invoke at each iteration. Determining the best strategy for selecting J will likely require some experimental study and could well be application-dependent. One can imagine greedy strategies in which one attempts to first select oracles that seem likely to be quick to evaluate, or to

yield a large increase in the objective estimate, or some combination of these criteria.

If U^i in (23) is a compact set, c^i is a continuous function, and errors $\eta^i \geq 0$ chosen in Algorithm 3 are bounded, then the overall error of the lower oracle scheme above is bounded regardless of the size of the index set I_{k+1}, [60, Lemma 2.5]. Thus, Algorithm 2 can possibly be employed with Algorithm 3 to explore additive structures in difficult optimization problems.

The recent work [78] proposes a more general family of incremental bundle methods for problems with additive structure. The methods in [78] require two additional assumptions, that are satisfied in many relevant applications. One is that the Lipschitz constant of the functions must be available. The other is that oracles must also produce upper estimates on the function values. Exploiting upper estimates is an interesting feature that allows to approximate function values at points where the oracle had not been called. Furthermore, the methods in [78] can skip oracle calls entirely for some of the component functions, not only at null steps as in [60] (and Algorithm 2 above combined with Algorithm 3), but also at serious steps. These methods can work with oracles whose error bound is either known or unknown. Furthermore, such oracles may not necessarily be able to attain arbitrary precision requested by the user/algorithm, or even provide reliable upper bounds. Of course, when dealing with such general oracles, some properties of the incremental algorithm might be lost (e.g., exactness of the computed solution).

4.4.3 Nonlinearly constrained problems

We next comment briefly on inexact proximal bundle methods for nonlinearly constrained problems of the form (1) where, mostly for simplicity, we consider that $G = \mathbb{R}^n$.

If the Slater condition holds, i.e., $h(\mathbf{z}) < 0$ for some $\mathbf{z} \in \mathbb{R}^n$, then the *exact penalty* approach can be used. Recall that the Slater condition ensures that the set of Lagrange multipliers associated to any solution of (1) is nonempty and bounded [73]. Then, for any ρ greater than the largest Lagrange multiplier associated to some solution to (1), in principle, problem (1) can be solved minimizing the exact penalty function

$$F(\mathbf{x}) := f(\mathbf{x}) + \rho \max\{h(\mathbf{x}), 0\},$$

i.e., the original problem is reduced to the unconstrained setting (17), with f replaced by F. Note, however, that Lagrange multipliers are obviously unknown (just like the solution itself), and thus a suitable value of the penalty parameter ρ is unkown as well. It thus needs to be adjusted along iterations, using appropriate strategies.

A more general approach is presented in [72], where a bilevel problem is considered (i.e., that of minimizing one function over the set of minimizers of another). The usual constrained optimization case is obtained taking the lower-level function as a penalization of infeasibility, e.g., $\max\{h(\mathbf{x}), 0\}$. The method in [72] does not require constraint qualifications, though "penalization" is not exact. In [72] exact oracles are assumed, but it should be possible to introduce inexactness along of nowadays well understood patterns.

Approximations of the constrained problem by bundle-like linearly constrained subproblems introduces an *exogenous* inaccuracy that can be easily handled together with the genuine errors in the f- and h-oracles. With a model of the form $\hat{F}_k(\mathbf{x}) := \hat{f}_k(\mathbf{x}) + \rho \max\{\hat{h}_k(\mathbf{x}), 0\}$, the property that $\hat{F}_k(\cdot) \leq F(\cdot) + \eta_F$ (for some overall error $\eta_F \geq 0$) follows from the f- and h-models and oracles. As already noted, the difficulty lies in estimating the penalty parameter. Also, in the inexact case, it appears that a more accurate F-oracle is needed. A possible update is $\rho_k = \max\{\rho_{k-1}, \lambda_k + 1\}$, where $\lambda_k \geq 0$ is a Lagrange multiplier associated to the solution $\mathbf{x}_{k+1}$ of the QP

$$\begin{cases} \text{minimize} & \hat{f}_k(\mathbf{x}) + \frac{1}{2t_k}\|\mathbf{x} - \hat{\mathbf{x}}_k\|_2^2 \\ \text{subject to} & \hat{h}_k(\mathbf{x}) \leq 0. \end{cases}$$

This QP is feasible (by the Slater assumption), and its solution also solves

$$\text{minimize} \quad \hat{f}_k(\mathbf{x}) + \rho_k \max\{\hat{h}_k(\mathbf{x}),\, 0\} + \frac{1}{2t_k}\|\mathbf{x} - \hat{\mathbf{x}}_k\|_2^2\,,$$

for sufficiently large ρ_k. For the approach to work as the exact penalty method, ρ_k needs to stabilize eventually, which requires the Lagrange multiplier sequence (λ_k) to be bounded, e.g., [63, Lemma 7.1]. Once the penalty parameter stabilizes at a value ρ, Theorem 4 applies to the function F. In particular, accumulation points of the sequence $(\hat{\mathbf{x}}_k)$ solve the constrained problem within an accuracy bound depending also on the asymptotic error made when estimating the penalty parameter at serious steps.

A specialized inexact proximal bundle method that does not require penalization is proposed in [3]. Therein, the authors deal with nonlinearly constrained convex programs by employing an improvement function that is slightly more general than the one considered in Section 3.

5 Doubly stabilized bundle method

Consider problem (1) with no h-constraint. As seen in the previous sections, level bundle methods employ the model $\hat{f}_k$ of f in the subproblem's constraints:

$$\begin{cases} \text{minimize} & \|\mathbf{x}-\hat{\mathbf{x}}_k\|_2^2 \\ \text{subject to} & \mathbf{x}\in \text{Ł}_k := \{\mathbf{x}\in G \mid \hat{f}_k(\mathbf{x}) \le f_k^{lev}\}. \end{cases} \tag{27}$$

The proximal bundle methods use the model in the the objective function of the subproblem:

$$\begin{cases} \text{minimize} & \hat{f}_k(\mathbf{x}) + \frac{1}{2t_k}\|\mathbf{x}-\hat{\mathbf{x}}_k\|_2^2 \\ \text{subject to} & \mathbf{x}\in G. \end{cases} \tag{28}$$

It is worth to mention that the choice of the parameter t_k in the proximal variant is quite a delicate task. Although the simplest choice $t_k = t > 0$ (for all k) is enough to prove theoretical convergence, it is well understood that for practical efficiency t_k must be properly updated along iterations. For level bundle algorithms a fixed level parameter f_k^{lev} is not possible as this may give infeasible level sets. But as seen in Section 3, there exist strategies that manage f_k^{lev} by simple explicit calculations. To simultaneously compute trial points and obtain a meaningful update for the prox-parameter t_k, the *doubly stabilized bundle method* - DSBM - of [14] combines level and proximal regularizations by adding to (28) the level constraint. More specifically, the method defines trial points by solving

$$\begin{cases} \text{minimize} & \hat{f}_k(\mathbf{x}) + \frac{1}{2t_k}\|\mathbf{x}-\hat{\mathbf{x}}_k\|_2^2 \\ \text{subject to} & \mathbf{x}\in G \ \text{ and } \ \hat{f}_k(\mathbf{x}) \le f_k^{lev}. \end{cases} \tag{29}$$

We first observe that the above doubly stabilized subproblem can be represented by

$$\begin{cases} \text{minimize} & r + \frac{1}{2t_k}\|\mathbf{x}-\hat{\mathbf{x}}_k\|_2^2 \\ \text{subject to} & f_{\mathbf{x}_j} + \langle \boldsymbol{\xi}_{\mathbf{x}_j}, \mathbf{x}-\mathbf{x}_j\rangle \le r, \quad j\in\mathcal{J}_k, \\ & \mathbf{x}\in G\,,\ r\in\mathbb{R} \ \text{ and } \ r \le f_k^{lev}. \end{cases}$$

This QP has just one extra scalar bound constraint compared to the QP resulting from (28), and one more variable compared to (27). Thus, (29) is no harder (or at least, cannot be much harder) to solve than (28) or (27).

Overall, there seems to be some consensus that for solving unconstrained problems proximal bundle methods are very good choices (though updating t_k is not straightforward), while for constrained problems level bundle methods might be preferable. The doubly stabilized bundle method combines the attractive features of both approaches in a single algorithm [14]. One advantage when compared to the proximal approach is that the level set constraint provides a certain Lagrange multiplier, which is used to update the proximal parameter in a simple manner. When compared to level bundle methods, the objective function $\hat{f}_k(\mathbf{x}) + \frac{1}{2t_k}\|\mathbf{x}-\hat{\mathbf{x}}_k\|_2^2$ with proximal regularization allows

for searching for good points inside of the level set Ł_k of (27), and not only on its boundary, as the level method does.

Another interesting feature of the doubly stabilized bundle method is that exact and inexact data are handled in the same manner, as is the case for the level variant and in contrast to the proximal.

5.1 Optimality certificate and more

We first state some properties of the minimizer $\mathbf{x}_{k+1}$ in (29).

Proposition 5. *If $L_k = \{x \in G \mid \hat{f}_k(\mathbf{x}) \leq f_k^{lev}\} \neq \emptyset$ then problem* (29) *has the unique solution $\mathbf{x}_{k+1}$. In addition, if G is polyhedral or $\mathtt{ri}\, G \cap \{\mathbf{x} \in \mathbb{R}^n : \hat{f}_k(\mathbf{x}) \leqslant f_k^{lev}\} \neq \emptyset$ then there exist $\mathbf{p}_k^f \in \partial \hat{f}_k(\mathbf{x}_{k+1})$ and $\mathbf{p}_k^G \in \partial i_G(\mathbf{x}_{k+1})$, and (scalar) Lagrange multipliers $\mu_k \geqslant 1$ and $\lambda_k \geqslant 0$ such that*

$$\mathbf{x}_{k+1} = \hat{\mathbf{x}}_k - t_k \mu_k \hat{\boldsymbol{\xi}}_k, \quad \textit{with} \;\; \hat{\boldsymbol{\xi}}_k = \mathbf{p}_k^f + \frac{1}{\mu_k}\mathbf{p}_k^G, \tag{30}$$

$$\mu_k = \lambda_k + 1 \;\; \textit{and} \;\; \lambda_k(\hat{f}_k(\mathbf{x}_{k+1}) - f_k^{lev}) = 0.$$

Furthermore, the aggregate linearization $\bar{f}_{k_a}(\cdot) := \hat{f}_k(\mathbf{x}_{k+1}) + \langle \hat{\boldsymbol{\xi}}_k, \cdot - \mathbf{x}_{k+1}\rangle$ satisfies $\bar{f}_{k_a}(\mathbf{x}) \leqslant \hat{f}_k(\mathbf{x}) \leqslant f(\mathbf{x}) + i_G(\mathbf{x}) + \eta$ for all $\mathbf{x} \in \mathbb{R}^n$, where we assume that (2) *holds for the oracle.*

Proof. See [14, Proposition 1].

Interestingly, the Lagrange multiplier μ_k in Proposition 5 indicates that the solution $\mathbf{x}_{k+1}$ of (29) either solves the proximal master problem (28), or the level one (27).

Lemma 3. *For $t_k > 0$ and $f_k^{lev} \in \mathbb{R}$, let $\mathbf{x}^{prox} \in G$ and $\mathbf{x}^{lev} \in G$ be the (unique) solutions of problems* (28) *and* (27)*, respectively. Let $\mathbf{x}_{k+1} \in G$ be the unique solution of problem* (29)*. Then it holds that*

$$\mathbf{x}_{k+1} = \begin{cases} \mathbf{x}^{prox} & \textit{if } \mu_k = 1, \\ \mathbf{x}^{lev} & \textit{if } \mu_k > 1, \end{cases}$$

where μ_k is the Lagrange multiplier defined in Proposition 5.

Proof. The result follows by comparing the optimality conditions of the three strongly convex subproblems (28), (27) and (29). See [14, Lemma 1] for full details. □

It is thus clear that each iteration of the doubly stabilized algorithm makes either a step of the associated proximal bundle method, or of the level method. At every iteration, the algorithm makes this choice *automatically.*

Once the iterate $\mathbf{x}_{k+1}$ is computed, the oracle provides the new estimate function value $f_{\mathbf{x}_{k+1}}$. As in the proximal bundle methods, we shall change the stability center when the descent test (20) is verified:

$$f_{\mathbf{x}_{k+1}} \leqslant f_{\hat{\mathbf{x}}_k} - \kappa v_k \,.$$

As before, $\kappa \in (0,1)$ and $v_k = f_{\hat{\mathbf{x}}_k} - \hat{f}_k(\mathbf{x}_{k+1})$ is the decrease predicted by the model. It is worth to mention that the level parameter f_k^{lev} also provides *expected* decrease

$$v_k^{lev} := f_{\hat{\mathbf{x}}_k} - f_k^{lev} > 0,$$

where the inequality follows from the fact that level parameters are chosen to satisfy $f_k^{lev} < f_{\hat{\mathbf{x}}_k}$ for all k. The following result establishes helpful connections among the predicted decrease v_k, the expected decrease v_k^{lev} and the aggregate error

$$\hat{e}_k = f_{\hat{\mathbf{x}}_k} - \bar{f}_{k_a}(\hat{\mathbf{x}}_k).$$

Proposition 6. *It holds that*

$$v_k = \hat{e}_k + t_k\mu_k\|\hat{\boldsymbol{\xi}}_k\|_2^2 \geqslant v_k^{lev}\,,$$

where μ_k is the Lagrange multiplier defined in Proposition 5. Moreover, if $\mu_k > 1$ then $v_k = v_k^{lev}$. Furthermore, it holds that

$$\hat{\boldsymbol{\xi}}_k \in \partial_{(\hat{e}_k+2\eta)}[f(\hat{\mathbf{x}}_k) + i_G(\hat{\mathbf{x}}_k)].$$

Proof. It follows from the definition of $\hat{e}_k$ and the left-most formula in (30) that

$$\hat{e}_k = f_{\hat{\mathbf{x}}_k} - \bar{f}_{k_a}(\hat{\mathbf{x}}_k) = f_{\hat{\mathbf{x}}_k} - [\hat{f}_k(\mathbf{x}_{k+1}) + \langle\hat{\boldsymbol{\xi}}_k, \hat{\mathbf{x}}_k - \mathbf{x}_{k+1}\rangle] = v_k - t_k\mu_k\|\hat{\boldsymbol{\xi}}_k\|_2^2.$$

In addition, since $\mathbf{x}_{k+1}$ is feasible in (29), we have that $\hat{f}_k(\mathbf{x}_{k+1}) \leqslant f_k^{lev} = f_{\hat{\mathbf{x}}_k} - v_k^{lev}$, which implies $v_k^{lev} \leqslant v_k$. Recall also that if $\mu_k > 1$ then $\lambda_k > 0$, in which case (30) implies $\hat{f}_k(\mathbf{x}_{k+1}) = f_k^{lev}$, so that $v_k^{lev} = v_k$. The inclusion $\hat{\boldsymbol{\xi}}_k \in \partial_{(\hat{e}_k+2\eta)}[f(\hat{\mathbf{x}}_k) + i_G(\hat{\mathbf{x}}_k)]$ follows from the same steps in the proof of Lemma 2. □

As in the proximal bundle method, we can stop the method when both $\hat{e}_k$ and $\hat{\boldsymbol{\xi}}_k$ are small enough. An alternative stopping test is borrowed from the level variant: the management of the level parameter f_k^{lev} provides (inexact) lower bound f_k^{low} and thus DSBM can stop when $f_{\hat{\mathbf{x}}_k} - f_k^{low}$ is small enough.

5.2 Doubly stabilized bundle algorithm

We now present the inexact doubly stabilized algorithm of [14] for convex problems in the form (17).

Algorithm 4: Doubly Stabilized Bundle Method

Data: Stopping tolerances $\delta_1^{tol}, \delta_2^{tol}, \delta_3^{tol} \geq 0$, descent parameters $\kappa, \gamma \in (0,1)$, a stepsize $t_0 \geq \underline{t} > 0$, $\tau \in [\frac{1}{2}, 1)$ and $v_0^{lev} > 0$.

Step 0 (*Initialization*) Choose a starting point $\mathbf{x}_0 \in G$, set $\hat{\mathbf{x}}_0 := \mathbf{x}_0$, and call the oracle to compute $(f_{\mathbf{x}_0}, \boldsymbol{\xi}_{\mathbf{x}_0})$. Set $\mathcal{J}_0 := \{0\}$ and $k := 0$.

Step 1 (*First stopping criterion*) Set $\Delta_k := f_{\hat{\mathbf{x}}_k} - f_k^{low}$. If $\Delta_k \leq \delta_3^{tol}$, stop.

Step 2 (*Trial point computation*) Set $f_k^{lev} := f_{\hat{\mathbf{x}}_k} - v_k^{lev}$ and try to solve (29) to obtain $\mathbf{x}_{k+1}$ and a Lagrange multiplier λ_k associated to the level constraint $\hat{f}_k(\mathbf{x}) \leqslant f_k^{lev}$. Set $\mu_k := \lambda_k + 1$, $\hat{\boldsymbol{\xi}}_k := (\hat{\mathbf{x}}_k - \mathbf{x}_{k+1})/(t_k \mu_k)$ and $\hat{e}_k := v_k - t_k \mu_k \|\hat{\boldsymbol{\xi}}_k\|_2^2$. If (29) is infeasible set $f_k^{low} := f_k^{lev}$, $v_k^{lev} := (1-\gamma)\Delta_k$ and go back to Step 1.

Step 3 (Second stopping criterion) If $\|\boldsymbol{\xi}_k\|_2 \leq \delta_1^{tol}$ and $\hat{e}_k \leq \delta_2^{tol}$, stop.

Step 4 (Oracle call and descent test) Call the inexact oracle (2) to compute $(f_{\mathbf{x}_{k+1}}, \boldsymbol{\xi}_{\mathbf{x}_{k+1}})$. Set $f_{k+1}^{low} = f_k^{low}$.

- (Serious step) If the descent test (20) holds, then set $\hat{\mathbf{x}}_{k+1} := \mathbf{x}_{k+1}$, $t_{k+1} := \mu_k t_k$ and $v_{k+1}^{lev} := \min\{v_k^{lev}, (1-\gamma)(f_{\hat{\mathbf{x}}_{k+1}} - f_{k+1}^{low})\}$.
- (Null step) Otherwise, set $\hat{x}_{k+1} := \hat{x}_k$. If $\mu_k > 1$ (level iterate) and $\hat{e}_k \geq -\tau t_k \mu_k \|\hat{\boldsymbol{\xi}}_k\|_2^2$, set $v_{k+1}^{lev} := \gamma v_k^{lev}$; otherwise set $v_{k+1}^{lev} := v_k^{lev}$. Choose $t_{k+1} \in [\underline{t}, t_k]$.

Step 5 (Bundle management) Same as Step 5 of Algorithm 2. Increase k by 1 and go to Step 1.

Observe that the lower bound f_k^{low} is only updated when the level set L_k is empty in Step 2. It thus follows from a similar analysis to the one presented in Lemma 1 that $f_k^{low} \leq f^* + \eta$ for all k as long as f_0^{low} is a valid lower bound. (Note that Algorithm 4 accepts the choice that $f_0^{low} := -\infty$.) Therefore, if the algorithm stops at Step 1, we have that

$$\delta_3^{tol} \geq f_{\hat{\mathbf{x}}_k} - f_k^{low} \geq f(\hat{\mathbf{x}}_k) - \eta - (f^* + \eta)\,,$$

i.e., $\hat{\mathbf{x}}_k$ is a $(\delta_3^{tol} + 2\eta)$-approximate solution to problem (17).

We point out that $v_k^{lev} > 0$ as long as $\Delta_k > 0$. This property with Proposition 6 ensures that $v_k > 0$ for all k, i.e., the decent test (20) makes sense always. This is why Algorithm 4 does not need any additional procedure to handle inexact data, such as *noise attenuation* in Algorithm 2.

Step 5 increases the proximal parameter t_k only after descent steps resulting from level iterations ($\mu_k > 1$). On the other hand, t_k can be decreased only after null steps. A simple rule used in the numerical experiments of [14] is $t_{k+1} := \max\{\underline{t}, t_k v_k^{lev}/v_k\}$, which decreases the proximal parameter only after null steps resulting from proximal iterations ($v_k > v_k^{lev}$ is only possible when $\mu_k = 1$, see Proposition 6). In this manner, the level parameter f_k^{lev} and the multiplier μ_k indicate how to update the proximal parameter t_k. This simple strategy has shown good performance in practice.

It is interesting to note that if at Step 2 (for all k) one sets $f_k^{lev} = \infty$, Algorithm 4 becomes a proximal bundle algorithm, c.f. Lemma 3.

5.3 Convergence analysis

Convergence analysis of the doubly stabilized bundle method has to account for all the possible combinations of level and proximal steps, whether null or descent, and the possibility of empty level sets. To that end the following three possible cases are considered: (i) the level sets Ł_k are empty infinitely many times; (ii) item (i) does not happen, and infinitely many descent steps are generated; (iii) in the same situation, finitely many descent steps are generated. In what follows, we assume that $\delta_1^{tol} = \delta_2^{tol} = \delta_3^{tol} = 0$ and that Algorithm 4 does not stop. If the algorithm stops for zero tolerances in Step 1 or Step 3, then the last descent iterate is a 2η-solution to the problem, as previously mentioned.

Proposition 7 (Infinitely many empty level sets). *Suppose the level set Ł_k is empty infinitely many times. Then $\Delta_k \to 0$.*

Proof. Every time Ł_k is found to be empty, the lower bound is increased by $v_k^{lev} > 0$. Since $f^* + \eta \geq f_k^{low}$ for all k, then $v_k^{lev} \to 0$. The definition $v_k^{lev} := (1-\gamma)\Delta_k$ (and monotonicity of $(f_{\hat{\mathbf{x}}_k})$) yields $\Delta_k \to 0$. □

Consider now the case where $\text{Ł}_k \neq \emptyset$ for all k large enough, and there is a finite number of descent steps.

Proposition 8 (Infinite loop of null steps). *If the algorithm generates an infinite sequence of null steps after a last serious iterate, then* (21) *holds with $\mathcal{K} = \{0, 1, \ldots\}$.*

The proof of Proposition 8 is rather technical; see [14, Lemma 7].

Proposition 9 (Infinitely many serious steps). *If the algorithm generates an infinite sequence of serious steps, then $\Delta_k \to 0$ or/and* (21) *holds for $\mathcal{K} = \{k \in \mathbb{N} \mid$ the descent test* (20) *is satisfied for $k\}$.*

Proof. Analogously to the proof of Proposition 4, one obtains that $v_k \to 0$ as $\mathcal{K} \ni k \to \infty$. Proposition 6 then ensures that $v_k^{lev} \to 0$ as $\mathcal{K} \ni k \to \infty$. If there exists an infinite subset $\mathcal{K}' \subset \mathcal{K}$ such that

$$v_{k+1}^{lev} = \min\{v_k^{lev}, (1-\gamma)(f_{\hat{\mathbf{x}}_{k+1}} - f_{k+1}^{low})\}$$

for all $k \in \mathcal{K}'$ in Step 2, then $\Delta_k \to 0$ as $\mathcal{K}' \ni k \to \infty$, as well as for $\mathcal{K} \ni k \to \infty$. Otherwise, v_k^{lev} is decreased during null steps. In the latter case, the condition $\hat{e}_k \geq -\tau t_k \mu_k \|\hat{\boldsymbol{\xi}}_k\|_2^2$ holds. Then, as in the proof of Proposition 3, $v_k \to 0$ as $\mathcal{K} \ni k \to \infty$ implies (21). □

Combining all the cases considered above, we conclude the following.

Theorem 5. *Suppose that problem* (17) *has bounded level sets, the inexact oracle satisfies* (2) *and that in Algorithm 4 one sets* $\delta_1^{tol} = \delta_2^{tol} = \delta_3^{tol} = 0$. *Then* $\Delta_k \to 0$ *or/and the optimality certificate* (21) *holds for some index set* $\mathcal{K}$. *Furthermore, every accumulation point* $\bar{\mathbf{x}}$ *of the sequence* $(\hat{\mathbf{x}}_k)$ *is a* 2η*-solution to problem* (17).

6 Nonconvex objective functions

Consider the problem

$$\begin{cases} \text{minimize} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in G, \end{cases} \tag{31}$$

where the feasible set $G \subset \mathbb{R}^n$ is convex and compact, and the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is *proper* [66, p. 5], *regular* [66, Definition 7.25], and locally Lipschitz-continuous with full domain. Note that f need not be convex. In this situation, the Clarke subdifferential $\partial f(\mathbf{x})$ is well-defined for $\mathbf{x} \in \mathbb{R}^n$, and it can be considered as the convex hull of all possible limits of gradients at points of differentiablity, for all sequences of such points converging to $\mathbf{x}$. For alternative definitions of the subdifferential of regular functions, see [66, Chapter 8].

Again, we are interested in the situations where for a given point, only some inexact information about the function and subgradient values is available. For function values, this is modeled the same way as in the convex case above: for each given $\mathbf{x} \in \mathbb{R}^n$ an oracle returns some estimate $f_{\mathbf{x}}$ such that

$$f_{\mathbf{x}} = f(\mathbf{x}) - \eta_{\mathbf{x}}^v, \tag{32}$$

where the sign of errors $\eta_{\mathbf{x}}^v \in \mathbb{R}$ is not specified (so that the true function value can be either overestimated or underestimated).

As for approximate subgradients, we note that in the nonconvex case, a number of different interpretations of inexactness is possible. Here, we adopt the rather natural view of, e.g., [29, 74]. In particular, we consider that at a point $\mathbf{x} \in \mathbb{R}^n$, an element $\boldsymbol{\xi}_{\mathbf{x}} \in \mathbb{R}^n$ approximates within tolerance $\eta_{\mathbf{x}}^s \geq 0$ some subgradient of f at $\mathbf{x}$ if

$$\boldsymbol{\xi}_{\mathbf{x}} \in \partial f(\mathbf{x}) + \bar{B}(\mathbf{0}; \eta_{\mathbf{x}}^s), \tag{33}$$

where $\bar{B}(\mathbf{0}; \eta_{\mathbf{x}}^s)$ is the closed (Euclidean) ball of radius $\eta_{\mathbf{x}}^s$ centered at the origin. This means that there exists some exact subgradient $\mathbf{z} \in \partial f(\mathbf{x})$ such that $\|\boldsymbol{\xi}_{\mathbf{x}} - \mathbf{z}\|_2 \leq \eta_{\mathbf{x}}^s$, i.e., $\boldsymbol{\xi}_{\mathbf{x}}$ approximates *this* subgradient $\mathbf{z}$ with the tolerance $\eta_{\mathbf{x}}^s \geq 0$.

The error terms in (32) and (33) are assumed to be uniformly bounded on the feasible set: for all $\mathbf{x} \in G$, it holds that

$$|\eta^v_{\mathbf{x}}| \leq \eta^v \quad \text{and} \quad 0 \leq \eta^s_{\mathbf{x}} \leq \eta^s. \tag{34}$$

However, the error terms themselves, or even their bounds η^v and η^s, are usually unknown.

6.1 Some examples of inexact nonconvex oracles

Next, we briefly discuss some situations which naturally lead to the setting of (32), (33) and (34).

Example 6 (Derivative-free optimization). Suppose f is twice continuously differentiable (generally nonconvex), but only its (exact) function values are accessible, with no derivatives information available. This is the framework of *derivative-free optimization*; see [6, 12]. In this setting, the gradients are approximated by finite differences, linear interpolation, or other techniques. Suitable error bounds can be given for a good number of such techniques; see [12, § 2–5] for some examples. Similar error bounds exist also for a variety of other (sub-)gradient approximation methods [7, 25, 26, 27, 42]. In general, these error bounds involve the Lipschitz constant of the true gradient, the geometry of the sample set of points used to create the approximation, and the diameter of the sample set. As the sample set is created by the user, its geometry and diameter are controlled. The compactness of G, in turn, yields a bound on the Lipschitz constant. One has then the error bound for the approximated gradients, but the value of the bound is unknown, of course. □

Example 7 (H_∞-control). In H_∞-control [4, 56], certain nonconvex functions can be locally approximated by using the support function of a compact set. A detailed explanation of this approximation technique is given in [56, § 1.9]. An error bound of the form (32), (33) and (34), is provided in [4, Lemma 2.1]. Moreover, the function in question is lower-$\mathcal{C}^2$ (and therefore locally Lipschitz) [56, Lemma 9]. □

Example 8 (Stochastic simulations). When the objective function is provided through stochastic simulation, the errors in the function and subgradient evaluations are understood through probability distribution functions. This encompasses Example 5 above, where also some details are given. □

6.2 Models: handling nonconvexity and inexactness

Bundle methods for nonconvex functions using *exact* information have been considered in [4, 22, 28, 36, 41, 46, 47, 50, 77]. Note that because of nonconvexity, even when exact function and subgradient values are used to build the cutting-plane model of f, this model can cut the graph of the function (and thus its minima). In other words, unlike in the convex case, the so-called *linearization errors* can be negative, even when the data is exact. To deal with this issue, most methods "downshift" the model, i.e., negative linearization errors are made nonnegative by "brute force" of downshifting. The method of [28] also tilts the slopes of the cutting planes, in addition to downshifting.

In the nonconvex case with *inexact* oracles, negative linearization errors may be caused by nonconvexity of the function, by inexact data, or by both. This presents additional challenges. To the best of our knowledge, the only works which deal with inexact information in bundle methods for nonconvex functions are [29] and [56]. The method of [56] employs the downshift mechanism that modifies linearization errors if they are negative. The method of [29] uses the ideas of [28]; in particular, it both downshifts the cutting-planes and tilts their slopes. We next outline the algorithm of [29].

As in the usual proximal bundle methods, at iteration k one has available information computed at some previous iterations: in our case, $f_{\mathbf{x}_j}$ and $\boldsymbol{\xi}_{\mathbf{x}_j}$, $j \in \mathcal{J}_k$ ($j \leq k$), which are the approximate function and subgradient values satisfying the relations (32), (33), (34) written for $\mathbf{x} = \mathbf{x}_j$. Among the previous iterates, one is designated as the stability center $\hat{\mathbf{x}}_k \in G$ (the "best" point computed so far, with the approximate function value $f_{\hat{\mathbf{x}}_k}$).

Note that the usual cutting-plane model of the objective function f obtained from this information would be given by

$$\max_{j\in\mathcal{J}_k}\{f_{\mathbf{x}_j} + \langle \boldsymbol{\xi}_{\mathbf{x}_j}, \cdot - \mathbf{x}_j\rangle\} = f_{\hat{\mathbf{x}}_k} + \max_{j\in\mathcal{J}_k}\{-e_j^k + \langle \boldsymbol{\xi}_{\mathbf{x}_j}, \cdot - \hat{\mathbf{x}}_k\rangle\},$$

where

$$e_j^k = f_{\hat{\mathbf{x}}_k} - f_{\mathbf{x}_j} - \langle \boldsymbol{\xi}_{\mathbf{x}_j}, \hat{\mathbf{x}}_k - \mathbf{x}_j\rangle \tag{35}$$

are the linearization errors. These linearization errors would have been nonnegative in the convex case with exact data, but not in our setting. We thus introduce the following *modified* cutting-plane model:

$$\hat{f}_k(\mathbf{x}) := f_{\hat{\mathbf{x}}_k} + \max_{j\in\mathcal{J}_k}\{-c_j^k + \langle \mathbf{s}_j^k, \mathbf{x} - \hat{\mathbf{x}}_k\rangle\}, \tag{36}$$

where in each affine piece both the intercept c_j^k and the slope $\mathbf{s}_j^k$ correspond, respectively, to the linearization error and an approximate subgradient of the "locally convexified" function $f(\cdot) + \frac{\beta_k}{2}\|\cdot - \hat{\mathbf{x}}_k\|_2^2$. The convexification parameter $\beta_k > 0$ is adjusted dynamically along iterations, and is taken sufficiently large to make the intercept c_j^k nonnegative. Accordingly, each affine piece has a shifted nonnegative intercept

$$0 \leq c_j^k := e_j^k + b_j^k, \quad \text{where} \quad b_j^k := \frac{\beta_k}{2}\|\mathbf{x}_j - \hat{\mathbf{x}}_k\|_2^2, \tag{37}$$

with e_j^k given by (35), and a modified slope

$$\mathbf{s}_j^k := \boldsymbol{\xi}_{\mathbf{x}_j} + \beta_k(\mathbf{x}_j - \hat{\mathbf{x}}_k), \tag{38}$$

which results from tilting the given approximate subgradient $\boldsymbol{\xi}_{\mathbf{x}_j}$ at $\mathbf{x}_j$ by means of the convexification parameter β_k. Any choice of β_k that makes c_k^j in (37) nonnegative is acceptable; we take

$$\beta_k \geq \max\left\{\max_{j\in\mathcal{J}_k,\, \mathbf{x}_j\neq\hat{\mathbf{x}}_k} \frac{-2e_j^k}{\|\mathbf{x}_j - \hat{\mathbf{x}}_k\|_2^2}, 0\right\} + \gamma\,, \tag{39}$$

for a (small) positive parameter γ.

Once the cutting-plane model is set up, choosing a prox-parameter $t_k > 0$, the new iterate is given by $\mathbf{x}_{k+1}$, the solution of the usual subproblem of the proximal bundle method:

$$\begin{cases} \text{minimize} & \hat{f}_k(\mathbf{x}) + \frac{1}{2t_k}\|\mathbf{x} - \hat{\mathbf{x}}_k\|_2^2 \\ \text{subject to} & \mathbf{x} \in G\,. \end{cases} \tag{40}$$

6.3 Key relations and the stationarity certificate

We next discuss relations between the objects the algorithm constructs based on solving (40), with the model satisfying (36)–(39), and how these objects can be related to (approximate) stationarity for problem (31).

First, as (40) has the structure of the usual proximal bundle method subproblem, it holds (just as in the convex case), that

$$\mathbf{x}_{k+1} := \hat{\mathbf{x}}_k - t_k\hat{\boldsymbol{\xi}}_k, \quad \text{with} \quad \hat{\boldsymbol{\xi}}_k := \mathbf{p}_k^f + \mathbf{p}_k^G,$$

where $\mathbf{p}_k^G \in \partial i_G(\mathbf{x}_{k+1})$, $\mathbf{p}_k^f \in \partial\hat{f}_k(\mathbf{x}_{k+1})$, and

$$\mathbf{p}_k^f := \sum_{j\in\mathcal{J}_k} \alpha_j^k \mathbf{s}_j^k,\ \sum_{j\in\mathcal{J}_k} \alpha_j^k = 1,\ \alpha_j^k \geq 0\ j \in \mathcal{J}_k\,.$$

Then the *aggregate linearization* is given as previously, i.e., $\bar{f}_{k_a}(\mathbf{x}) := \hat{f}_k(\mathbf{x}_{k+1}) + \langle\hat{\boldsymbol{\xi}}_k, \mathbf{x} - \mathbf{x}_{k+1}\rangle$, and the *aggregate error* by

$$\hat{e}_k := \hat{f}_k(\hat{\mathbf{x}}_k) - \hat{f}_k(\mathbf{x}_{k+1}) - \langle\mathbf{p}_k^f, \hat{\mathbf{x}}_k - \mathbf{x}_{k+1}\rangle \geq 0, \tag{41}$$

where the inequality is by the fact that $\mathbf{p}_k^f \in \partial \hat{f}_k(\mathbf{x}_{k+1})$. It can be further seen that $f_{\hat{\mathbf{x}}_k} = \hat{f}_k(\hat{\mathbf{x}}_k)$, and that

$$\hat{e}_k = f_{\hat{\mathbf{x}}_k} - \bar{f}_{k_a}(\mathbf{x}_{k+1}) + \langle \mathbf{p}_k^f, \mathbf{x}_{k+1} - \hat{\mathbf{x}}_k \rangle = \sum_{j \in \mathcal{J}_k} \alpha_j^k c_j^k. \tag{42}$$

For the aggregate linearization, it holds that

$$\bar{f}_{k_a}(\mathbf{x}) = f_{\hat{\mathbf{x}}_k} + \sum_{j \in \mathcal{J}_k} \alpha_j^k (-c_j^k + \langle \mathbf{s}_j^k, \mathbf{x} - \hat{\mathbf{x}}_k \rangle) = f_{\hat{\mathbf{x}}_k} - \hat{e}_k + \langle \mathbf{p}_k^f, \mathbf{x} - \hat{\mathbf{x}}_k \rangle,$$

where we have used (42). The following result (Lemma 4 below) shows what the algorithm should aim for, in order to compute (approximate) stationary points of problem (31).

We note that the proof of Lemma 4 uses the following assumption: "The number of active indices, i.e., of $j \in \mathcal{J}_k$ such that $\alpha_j^k > 0$, is uniformly bounded in k". As a practical matter, this can be readily achieved if D is polyhedral (the typical case), and an active-set QP solver is employed to solve subproblems (40) (this is because active-set QP solvers choose linearly independent bases).

The last assertion in Lemma 4 refers to lower-$\mathcal{C}^1$ functions, introduced in [75]. It is a broad class of locally Lipschitz-continuous functions that contains lower-$\mathcal{C}^2$ functions. One of the equivalent characterizations of f being a lower-$\mathcal{C}^1$ function consists in f being semismooth (see, e.g., [49]) and regular.

Lemma 4. *Suppose the cardinality of the set $\{j \in \mathcal{J}_k \mid \alpha_j^k > 0\}$ is uniformly bounded in k. If $\hat{e}^k \to 0$ as $k \to \infty$ and, for some subset $\mathcal{K} \subset \{1, 2, \ldots\}$, $\hat{\mathbf{x}}_k \to \bar{\mathbf{x}}$, $\hat{\boldsymbol{\xi}}_k \to 0$ as $\mathcal{K} \ni k \to \infty$, with $(\beta_k \mid k \in \mathcal{K})$ bounded, then*

$$\mathbf{0} \in \partial f(\bar{\mathbf{x}}) + \partial i_G(\bar{\mathbf{x}}) + \bar{B}(\mathbf{0}; \eta^s). \tag{43}$$

If, in addition, f is lower-$\mathcal{C}^1$, then for each $\varepsilon > 0$ there exists $\rho > 0$ such that for all $\mathbf{y} \in G \cap \bar{B}(\bar{\mathbf{x}}; \rho)$, it holds that

$$f(\mathbf{y}) \geq f(\bar{\mathbf{x}}) - (\eta^s + \varepsilon)\|\mathbf{y} - \bar{\mathbf{x}}\|_2 - 2\eta^f. \tag{44}$$

Proof. See [29, Lemma 5]. □

The result above indicates that to find an approximate stationary point of problem (31), the algorithm should drive $\hat{e}^k$ and $\hat{\boldsymbol{\xi}}_k$ to zero along the iterations.

6.4 Inexact nonconvex proximal bundle algorithm

Naturally, $\hat{e}^k$ and $\|\hat{\boldsymbol{\xi}}_k\|_2$ are driven to zero by means of a sufficient descent condition for the objective function with respect to its value at the stability center (recall, however, that both are inexact values in our setting). Specifically, the new iterate $\mathbf{x}_{k+1}$ computed by solving (40) is accepted as the new stability center if

$$f_{\mathbf{x}_{k+1}} \leq f_{\hat{\mathbf{x}}_k} - \kappa v_k, \tag{45}$$

where $\kappa \in (0,1)$ and

$$v_k := \hat{e}^k + t_k \|\hat{\boldsymbol{\xi}}_k\|_2^2. \tag{46}$$

If (45) does not hold, then a null step is declared and the stability center is not changed.

Algorithm 5: Inexact Proximal Bundle Method for Nonconvex Functions

Data: Stopping tolerance $\delta^{tol} \geq 0$, a descent parameter $\kappa \in (0,1)$, convexification safeguarding parameter $\gamma > 0$.

Step 0 (*Initialization*) Choose a starting point $\mathbf{x}_0 \in G$, set $\hat{\mathbf{x}}_0 := \mathbf{x}_0$, and call the oracle to compute $(f_{\mathbf{x}_0}, \boldsymbol{\xi}_{\mathbf{x}_0})$. Set $\mathcal{J}_0 := \{0\}$, $k := 0$.

Step 1 (*Trial point computation*) Find $\mathbf{x}_{k+1}$ solving (40), and let α_j^k, $j \in \mathcal{J}_k$, denote the corresponding optimal simplicial Lagrange multipliers. Compute $\hat{\boldsymbol{\xi}}_k = (\hat{\mathbf{x}}_k - \mathbf{x}_{k+1})/t_k$, $\hat{e}_k$ by (42), and v_k by (46).

Step 2 (Stopping criterion) If $v_k \leq \delta^{tol}$, stop.

Step 3 (Oracle call and descent test) Call the inexact oracle to compute $(f_{\mathbf{x}_{k+1}}, \boldsymbol{\xi}_{\mathbf{x}_{k+1}})$. If the descent test (45) holds, then declare the iterate serious: set $\hat{\mathbf{x}}_{k+1} := \mathbf{x}_{k+1}$. Otherwise, declare the iterate null: set $\hat{x}_{k+1} := \hat{x}_k$.

Step 4 (Bundle management) Set $\mathcal{J}_{k+1} := \{j \in \mathcal{J}^k : \alpha_j^k \neq 0\} \cup \{k+1\}$.

Step 5 (Parameters updating and loop). If the iterate was declared serious, select $t_{k+1} > 0$. If the iterate was declared null, select $0 < t_{k+1} \leq t_k$.
Increase k by 1. Compute the convexification parameter by (39), and the new intercepts c_j^k and slopes $\mathbf{s}_j^k$ by (37) and (38), respectively.
Go to Step 1.

It is worth noting that, contrary to many nonconvex bundle methods endowed with a linesearch, e.g., [36, 41, 47], Algorithm 5 does not employ linesearch. In [29, Section 5] some explanations are given as to why linesearch is not required here.

6.5 Convergence results

Depending on the assumptions about the prox-parameters t_k, it can be proven that the approximate optimality conditions stated in Lemma 4 hold for some accumulation point of $(\hat{\mathbf{x}}_k)$; or for all accumulation points of this sequence; or for the last serious iterate generated. Naturally, it is assumed that $\delta^{tol} = 0$ and an infinite sequence $(\mathbf{x}_k)$ is generated. Convergence results below assume that the sequence of convexification parameters (β_k) is bounded. This had been shown true for the related algorithm with exact data [28]. In the inexact setting, no proof is available at this time. The numerical results in [29] indicate that the assumption is reasonable and appears to hold in computation.

Theorem 6 (Infinitely many serious iterates). *Suppose Algorithm 5 generates an infinite number of serious steps. Then $v_k \to 0$ as $k \to \infty$. Let the sequence (β_k) be bounded. If $\sum_{k=1}^{\infty} t_k = +\infty$, then as $k \to \infty$ it holds that $\hat{e}_k \to 0$, and there exist $\mathcal{K} \subset \{1, 2, \ldots\}$ and $\bar{\mathbf{x}}$ such that $\hat{\mathbf{x}}_k \to \bar{\mathbf{x}}$ and $\hat{\boldsymbol{\xi}}_k \to 0$ as $\mathcal{K} \ni k \to \infty$. In particular, if the cardinality of the set $\{j \in \mathcal{J}_k \mid \alpha_j^k > 0\}$ is uniformly bounded in k, then the conclusions of Lemma 4 hold for $\bar{\mathbf{x}}$. If $\liminf_{k\to\infty} t_k > 0$, then these assertions hold for all accumulation points of the sequence $(\hat{\mathbf{x}}_k)$.*

Proof. See [29, Theorem 6].

Theorem 7 (Finite serious steps followed by infinitely many null steps). *Suppose Algorithm 5 generates a finite number of serious iterates, the last being $\bar{\mathbf{x}} := \hat{\mathbf{x}}_{k_0}$, followed by infinite null steps. Let the sequence (β_k) be bounded and let $\liminf_{k\to\infty} t_k > 0$. Then $\mathbf{x}_k \to \bar{\mathbf{x}}$, $v_k \to 0$, $\hat{e}_k \to 0$ and $\hat{\boldsymbol{\xi}}_k \to 0$. In particular, if the cardinality of the set $\{j \in \mathcal{J}_k \mid \alpha_j^k > 0\}$ is uniformly bounded in k, then the conclusions of Lemma 4 hold for $\bar{\mathbf{x}}$.*

Proof. See [29, Theorem 7].

We conclude by noting that the convergence results for Algorithm 5 are quite similar to the ones for the algorithm in [56]. In the latter reference, the objective function f is either ε-convex or lower-$\mathcal{C}^1$, and bounded lower level sets for f are assumed (instead of boundedness of the feasible set G). Some details of comparisons between Algorithm 5 and [56] are given in [29, Section 5].

7 Concluding remarks and research perspectives

We see several possible directions to develop further or broaden optimization techniques discussed in this work.

One research direction would be extending level bundle methods to deal with nonconvex objective and/or constraint functions. While proximal bundle

methods have been adapted to deal with nonconvexity (both in the exact and inexact settings, see Section 6), no level or doubly stabilized variants exist for this setting.

Another topic worth to investigate is the handling of inexact oracles in optimization problems involving *Difference-of-Convex* (DC) functions. Although practical applications of DC programming involving hard-to-evaluate functions are known in the literature (see, e.g., [77, Section 5]), currently available bundle methods for DC programs [13, 23, 34] do not handle noisy oracles. Extending DC bundle methods to inexact data (with or without on-demand accuracy) is, in our opinion, a relevant topic for research. Besides, existent DC bundle methods are all of the proximal type. Investigating level bundle methods for DC programming may give rise to algorithms possessing dimension independent iteration complexity, both in the exact and inexact settings.

Finally, we close this chapter with some words and references on bundle methods software. Due to a large number of possible stability functions, rules to set parameters, descent and stopping tests, and oracle types, finding the most appropriate (inexact) bundle method for a given application is not a trivial task for a practitioner, especially when without very specific knowledge and skills. It therefore would be essential to make available open-source software (as generic as possible) implementing several variants of bundle methods. A practitioner could then choose, among many possibilities and upon trying some test problems, the most suitable bundle algorithm and its parameter settings, to solve her/his problem. Admittedly, there is a number of bundle software packages that have been around for a while by now. Still, making available a wide range of options is something that has not happened as of yet. That said, as many practical problems can be cast in specific classes of nonsmooth optimization problems such as Lagrangian relaxation, two-stage stochastic programs and dual decomposition of multistage stochastic programs, some functional and robust computational codes have been appearing relatively recently, in the last years. `Matlab` implementations of some of the algorithms discussed here are freely available on the first author's homepage. Other freely available implementations of bundle methods are:

- `C++` codes by Antonio Frangioni can be downloaded from his homepage http://pages.di.unipi.it/frangio;
- `Fortran` codes of a variety of bundle methods (convex, nonconvex, multiobjective, DC) by Napsu Karmitsa and her collaborators can be found at the link http://napsu.karmitsa.fi;
- `Julia` codes of several implementations of bundle methods (including the doubly stabilized bundle method discussed in Section 5) are available in the open-source and parallel package DSP https://github.com/Argonne-National-Laboratory/DSP, by Kibaek Kim and Victor M. Zavala.

Acknowledgements

The authors acknowledge Wim van Ackooij and Antonio Frangioni for useful suggestions that helped improve both the contents and the presentation of this work. The first author acknowledges the partial financial support of PGMO (Gaspard Monge Program for Optimization and Operations Research) of the Hadamard Mathematics Foundation, through the project "Models for planning energy investment under uncertainty". The second author is supported in part by CNPq Grant 303724/2015-3, by FAPERJ Grant 203.052/2016, and by the Russian Foundation for Basic Research grant 19-51-12003 NNIOa.

References

1. van Ackooij, W., Cruz, J.B., de Oliveira, W.: A strongly convergent proximal bundle method for convex minimization in Hilbert spaces. Optimization **65**(1), 145–167 (2016)
2. van Ackooij, W., de Oliveira, W.: Level bundle methods for constrained convex optimization with various oracles. Computational Optimization and Applications **57**(3), 555–597 (2014)
3. van Ackooij, W., Sagastizábal, C.: Constrained bundle methods for upper inexact oracles with application to joint chance constrained energy problems. SIAM Journal on Optimization **24**(2), 733–765 (2014)
4. Apkarian, P., Noll, D., Prot, O.: A proximity control algorithm to minimize nonsmooth and nonconvex semi-infinite maximum eigenvalue functions. J. Convex Anal. **16**(3-4), 641–666 (2009)
5. Astorino, A., Frangioni, A., Fuduli, A., Gorgone, E.: A nonmonotone proximal bundle method with (potentially) continuous step decisions. SIAM Journal on Optimization **23**(3), 1784–1809 (2013)
6. Audet, C., Hare, W.: Derivative-free and blackbox optimization. Springer Series in Operations Research and Financial Engineering. Springer, Cham (2017)
7. Bagirov, A.M., Karasözen, B., Sezer, M.: Discrete gradient method: derivative-free method for nonsmooth optimization. J. Optim. Theory Appl. **137**(2), 317–334 (2008)
8. Ben-Tal, A., Nemirovski, A.: Non-Euclidean restricted memory level method for large-scale convex optimization. Math. Program. **102**, 407–456 (2005)
9. Bonnans, J., Gilbert, J., Lemaréchal, C., Sagastizábal, C.: Numerical Optimization. Theoretical and Practical Aspects. Universitext. Springer-Verlag, Berlin (2006). Second edition, xiv+490 pp.
10. Borghetti, A., Frangioni, A., Lacalandra, F., Nucci, C.: Lagrangian heuristics based on disaggregated bundle methods for hydrothermal unit commitment. IEEE Transactions on Power Systems **18**, 313–323 (2003)
11. Charnes, A., Cooper, W.: Chance-constrained programming. Management Science **6**, 73–79 (1959-1960)
12. Conn, A.R., Scheinberg, K., Vicente, L.N.: Introduction to Derivative-free Optimization, *MPS/SIAM Series on Optimization*, vol. 8. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA; Mathematical Programming Society (MPS), Philadelphia, PA (2009)
13. de Oliveira, W.: Proximal bundle methods for nonsmooth DC programming. Journal of Global Optimization (2019). DOI 10.1007/s10898-019-00755-4

14. de Oliveira, W., Solodov, M.: A doubly stabilized bundle method for nonsmooth convex optimization. Mathematical Programming **156**(1), 125–159 (2016)
15. de Oliveira, W., Tcheou, M.P.: An inertial algorithm for DC programming. Set-Valued and Variational Analysis (2018). DOI 10.1007/s11228-018-0497-0
16. Emiel, G., Sagastizábal, C.: Incremental-like bundle methods with application to energy planning. Computational Optimization and Applications **46**, 305–332 (2010)
17. Fábián, C.: Bundle-type methods for inexact data. Central European Journal of Operations Research **8**, 35–55 (2000)
18. Fábián, C.I., Wolf, C., Koberstein, A., Suhl, L.: Risk-averse optimization in two-stage stochastic models: Computational aspects and a study. SIAM Journal on Optimization **25**(1), 28–52 (2015)
19. Fischer, I., Gruber, G., Rendl, F., Sotirov, R.: Computational experience with a bundle approach for semidefinite cutting plane relaxations of max-cut and equipartition. Mathematical Programming **105**(2), 451–469 (2006)
20. Floudas, C.A.: Generalized Benders Decomposition, 2nd edn. Springer - Verlag (2009)
21. Frangioni, A.: Generalized bundle methods. SIAM Journal on Optimization **13**(1), 117–156 (2002)
22. Fuduli, A., Gaudioso, M., Giallombardo, G.: Minimizing nonconvex nonsmooth functions via cutting planes and proximity control. SIAM Journal on Optimization **14**(3), 743–756 (2004)
23. Gaudioso, M., Giallombardo, G., Miglionico, G., Bagirov, A.M.: Minimizing nonsmooth DC functions via successive DC piecewise-affine approximations. Journal of Global Optimization **71**, 37–55 (2018)
24. Genz, A., Bretz, F.: Computation of multivariate normal and t probabilities. No. 195 in Lecture Notes in Statistics. Springer, Dordrecht (2009)
25. Gupal, A.M.: A method for the minimization of almost differentiable functions. Kibernetika (Kiev) (1), 114–116 (1977)
26. Hare, W., Macklem, M.: Derivative-free optimization methods for finite minimax problems. Optim. Methods Softw. **28**(2), 300–312 (2013)
27. Hare, W., Nutini, J.: A derivative-free approximate gradient sampling algorithm for finite minimax problems. Computational Optimization and Applications **56**(1), 1–38 (2013)
28. Hare, W., Sagastizábal, C.: A redistributed proximal bundle method for nonconvex optimization. SIAM Journal on Optimization **20**(5), 2442–2473 (2010)
29. Hare, W., Sagastizábal, C., Solodov, M.: A proximal bundle method for nonsmooth nonconvex functions with inexact information. Computational Optimization and Applications **63**(1), 1–28 (2016)
30. Helmberg, C., Rendl, F.: A spectral bundle method for semidefinite programming. SIAM Journal on Optimization **10**(3), 673–696 (2000)
31. Hintermüller, M.: A proximal bundle method based on approximate subgradients. Computational Optimization and Applications **20**, 245–266 (2001)
32. Hiriart-Urruty, J., Lemaréchal, C.: Convex Analysis and Minimization Algorithms I, 2nd edn. No. 305 in Grundlehren der mathematischen Wissenschaften. Springer-Verlag (1996)
33. Hiriart-Urruty, J., Lemaréchal, C.: Convex Analysis and Minimization Algorithms II, 2nd edn. No. 306 in Grundlehren der mathematischen Wissenschaften. Springer-Verlag (1996)
34. Joki, K., Bagirov, A.M., Karmitsa, N., Mäkelä, M.M.: A proximal bundle method for nonsmooth DC optimization utilizing nonconvex cutting planes. Journal of Global Optimization **68**(3), 501–535 (2017)
35. Karmitsa, N., Gaudioso, M., Joki, K.: Diagonal bundle method with convex and concave updates for large-scale nonconvex and nonsmooth optimization. Optimization Methods and Software **34**(2), 363–382 (2019)

36. Kiwiel, K.: A linearization algorithm for nonsmooth minimization. Math. Oper. Res. **10**(2), 185–194 (1985)
37. Kiwiel, K.: Exact penalty functions in proximal bundle methods for constrained convex nondifferentiable minimization. Math. Programming **52**(2, Ser. B), 285–302 (1991)
38. Kiwiel, K.: A proximal bundle method with approximate subgradient linearizations. SIAM Journal on Optimization **16**(4), 1007–1023 (2006)
39. Kiwiel, K.C.: An aggregate subgradient method for nonsmooth convex minimization. Mathematical Programming **27**(3), 320–341 (1983)
40. Kiwiel, K.C.: Approximations in proximal bundle methods and decomposition of convex programs. Journal of Optimization Theory and Applications **84**, 529–548 (1995)
41. Kiwiel, K.C.: Restricted step and Levenberg-Marquardt techniques in proximal bundle methods for nonconvex nondifferentiable optimization. SIAM J. Optim. **6**(1), 227–249 (1996)
42. Kiwiel, K.C.: A nonderivative version of the gradient sampling algorithm for nonsmooth nonconvex optimization. SIAM J. Optim. **20**(4), 1983–1994 (2010)
43. Lemaréchal, C.: An extension of Davidon methods to nondifferentiable problems. Mathematical Programming Study **3**, 95–109 (1975)
44. Lemaréchal, C.: Lagrangian relaxation. In: Computational combinatorial optimization (Schloß Dagstuhl, 2000), *Lecture Notes in Comput. Sci.*, vol. 2241, pp. 112–156. Springer, Berlin (2001)
45. Lemaréchal, C., Nemirovskii, A., Nesterov, Y.: New variants of bundle methods. Math. Program. **69**(1), 111–147 (1995)
46. Lukšan, L., Vlček, J.: A bundle-Newton method for nonsmooth unconstrained minimization. Math. Programming **83**(3, Ser. A), 373–391 (1998)
47. Mäkelä, M.M., Neittaanmäki, P.: Nonsmooth optimization. Analysis and algorithms with applications to optimal control. World Scientific Publishing Co., Inc., River Edge, NJ (1992)
48. Malick, J., de Oliveira, W., Zaourar, S.: Uncontrolled inexact information within bundle methods. EURO Journal on Computational Optimization **5**(1), 5–29 (2017)
49. Mifflin, R.: An algorithm for constrained optimization with semismooth functions. Mathematics of Operations Research **2**, 191–207 (1977)
50. Mifflin, R.: A modification and extension of Lemarechal's algorithm for nonsmooth minimization. In: Sorensen D.C., Wets R.J.B. (eds) Nondifferential and Variational Techniques in Optimization. Mathematical Programming Studies (17), 77–90 (1982)
51. Mifflin, R.: A quasi-second-order proximal bundle algorithm. Mathematical Programming **73**(1), 51–72 (1996)
52. Mifflin, R., Sagastizábal, C.: A $\mathcal{VU}$-algorithm for convex minimization. Mathematical Programming **104**(2-3), 583–608 (2005)
53. Miller, S.: Inexact bundle method for solving large structured linear matrix inequalities. Ph.D. thesis, University of California, Santa Barbara, California (2001)
54. Montonen, O., Karmitsa, N., Mäkelä, M.M.: Multiple subgradient descent bundle method for convex nonsmooth multiobjective optimization. Optimization **67**(1), 139–158 (2018)
55. Nasri, A., Kazempour, S.J., Conejo, A.J., Ghandhari, M.: Network-constrained AC unit commitment under uncertainty: A Benders' decomposition approach. IEEE Transactions on Power Systems **31**(1), 412–422 (2016)
56. Noll, D.: Bundle method for non-convex minimization with inexact subgradients and function values. In: Computational and analytical mathematics, *Springer Proc. Math. Stat.*, vol. 50, pp. 555–592. Springer, New York (2013)

57. Noll, D., Apkarian, P.: Spectral bundle methods for non-convex maximum eigenvalue functions: first-order methods. Mathematical Programming **104**(2), 701–727 (2005)
58. de Oliveira, W.: Regularized optimization methods for convex MINLP problems. TOP **24**(3), 665–692 (2016)
59. de Oliveira, W.: Target radius methods for nonsmooth convex optimization. Operations Research Letters **45**(6), 659 – 664 (2017)
60. de Oliveira, W., Eckstein, J.: A bundle method for exploiting additive structure in difficult optimization problems. Tech. rep. (2015)
61. de Oliveira, W., Sagastizábal, C.: Bundle methods in the XXI century: A birds'-eye view. Pesquisa Operacional **34**(3), 647–670 (2014)
62. de Oliveira, W., Sagastizábal, C.: Level bundle methods for oracles with on demand accuracy. Optimization Methods and Software **29**(6), 1180–1209 (2014)
63. de Oliveira, W., Sagastizábal, C., Lemaréchal, C.: Convex proximal bundle methods in depth: a unified analysis for inexact oracles. Math. Prog. Series B **148**, 241–277 (2014)
64. Oliveira, W., Sagastizábal, C., Scheimberg, S.: Inexact bundle methods for two-stage stochastic programming. SIAM Journal on Optimization **21**(2), 517–544 (2011)
65. Ouorou, A.: A proximal cutting plane method using Chebychev center for nonsmooth convex optimization. Math. Program. **119**(2), 239–271 (2009)
66. Rockafellar, R.T., Wets, R.J.B.: Variational analysis, *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*, vol. 317. Springer-Verlag, Berlin (1998)
67. Ruszczyński, A.: Decomposition Methods, *Handbooks in Operations Research and Management Science*, vol. 10. Elsevier, Amsterdam (2003)
68. Sagastizábal, C.: Divide to conquer: decomposition methods for energy optimization. Math. Program. **134**(1, Ser. B), 187–222 (2012)
69. Sagastizábal, C.: On Lagrangian decomposition for energy optimization. In: Proceedings of the 8th International Congress on Industrial and Applied Mathematics, pp. 289–310. Higher Ed. Press, Beijing (2015)
70. Sagastizábal, C., Solodov, M.: An infeasible bundle method for nonsmooth convex constrained optimization without a penalty function or a filter. SIAM Journal on Optimization **16**(1), 146–169 (2005)
71. Solodov, M.V.: On approximations with finite precision in bundle methods for nonsmooth optimization. Journal of Optimization Theory and Applications **119**(1), 151–165 (2003)
72. Solodov, M.V.: A bundle method for a class of bilevel nonsmooth convex minimization problems. SIAM J. Optim. **18**(1), 242–259 (2007)
73. Solodov, M.V.: Constraint Qualifications. Wiley Encyclopedia of Operations Research and Management Science (2011). DOI 10.1002/9780470400531.eorms0978
74. Solodov, M.V., Zavriev, S.K.: Error stability properties of generalized gradient-type algorithms. J. Optim. Theory Appl. **98**(3), 663–680 (1998)
75. Spingarn, J.E.: Submonotone subdifferentials of Lipschitz functions. Trans. Amer. Math. Soc. **264**(1), 77–89 (1981)
76. T. Arnold R. Henrion, A.M., Vigerske, S.: A mixed-integer stochastic nonlinear optimization problem with joint probabilistic constraints. Pacific Journal of Optimization **10**(1), 5–20 (2014)
77. van Ackooij, W., de Oliveira, W.: Nonsmooth and nonconvex optimization via approximate difference-of-convex decompositions. Journal of Optimization Theory and Applications (2019). DOI 10.1007/s10957-019-01500-3
78. van Ackooij, W., Frangioni, A.: Incremental bundle methods using upper models. SIAM Journal on Optimization **28**(1), 379–410 (2018)

79. van Ackooij, W., Frangioni, A., de Oliveira, W.: Inexact stabilized Benders' decomposition approaches with application to chance-constrained problems with finite support. Computational Optimization and Applications **65**(3), 637–669 (2016)
80. van Ackooij, W., Henrion, R., Möller, A., Zorgati, R.: Joint chance constrained programming for hydro reservoir management. Optimization and Engineering **15**, 509–531 (2014)
81. Wolfe, P.: A method of conjugate subgradients for minimizing nondifferentiable functions. Math. Programming Stud. **3**, 145–173 (1975)